UNIVERSITY OF OSLO
Department of Informatics

TCP's Influence on Interactive Multimedia Traffic

Master Thesis

Bent J. Lorentzen

TCP's Influence on Interactive Multimedia Traffic

Bent J. Lorentzen

Contents

List of Figures

List of Tables

Preface

Interactive multimedia traffic is usually a stream of packets containing voice, video a combination of these. Because of the user interaction, there is a high vulnerability to latency, jitter and packet loss. In this thesis, we have investigated TCP's influence (FTP and HTTP traffic) on interactive multimedia traffic through numerous simulations in the network simulator NS-2. Our findings include what types and amounts of traffic are required to disrupt an interactive multimedia stream. We have also done a thorough investigation into what kind of changes to endpoint and network variables make the greatest difference in TCP's influence on this kind of traffic.

The changes made in the endpoints that affected TCP's influence on the VoIP traffic were the choice of transport protocol and the rate of the VoIP calls. Silence suppression made a little effect for the worse, i.e., it reduced the VoIP bandwidth consumption, but the received quality was reduced due to higher loss.

The change of queuing methods from tail-drop to RED made the packet loss for the VoIP call drop to almost zero. The increase in queue length improved on the packet loss, but it also introduced higher latency. The distance to the bottleneck proved to have no effect at all. Our simulations in the 100Mbps version of our network shows that our results are valid for higher bandwidth as well. Our results indicate that the network has to be quite badly congested before serious disruption occurs for the VoIP application.

Acknowledgments

I would like to express my gratitude to my supervisors Pål Halvorsen and Carsten Griwodz. Without their guidance, understanding and patience, this thesis would have been of a significantly lower quality.

To Simula Research Laboratory, thank you for providing such a good working environment.

To the guys at the lab, thank you for your good advice and for being so helpful.

Finally, I would like to thank my girlfriend for her wonderful support and for helping me with the proof reading of this thesis.

Chapter 1

Introduction

This thesis investigates the Transport Control Protocol's (TCP) influence on interactive multimedia traffic. For our investigation, we have chosen to focus on Voice over Internet Protocol (VoIP) as an example of multimedia traffic.

1.1 Background

The Internet was originally constructed to be a fault-tolerant means of communication, and although users and technology have changed, the goal is still the same. Today, millions of users worldwide rely on the Internet for their communication needs. In the last years multimedia traffic has become a more and more common part of the Internet traffic. People expect to listen to music, watch video, place phone calls and play games on the Internet. A lot of this traffic is to some degree based on the interaction of its users.

More and more people use a VoIP system to place their daily phone calls either with or without video. VoIP systems allow users to place calls to other VoIP users, as well as regular phones, through their Internet connection. The continuing growth in the number of VoIP users is mainly due to economics since the packet switched Internet is a more cost efficient way of placing calls than the legacy circuit switched telephone network. VoIP has become big business and there are many providers. Cisco systems inc. [3], AT&T [2] and Telenor AS [9] are examples of such providers. The VoIP systems

range from simple applications for your home computer, to fully functional phones that are connected directly to your Internet connection.

Applications such as standardized VoIP and proprietary Skype [8] have grown quickly and continue to do so. There are no indications that this trend will change, and why should it? Humans like to communicate with each other, and for the first time in history distance is no longer the obstacle it used to be.

The motivation for this thesis is to show how vulnerable multimedia traffic in a packet switched network can be, and hopefully to find some ways to improve this. We also hope to stimulate further research in this field.

1.2 Problem Statement

There is now a wide variety of applications that make use of VoIP. In addition to VoIP telephone systems, it is often integrated in Multi-player online games and used for phone and video conferences. Few of these applications adapt to bandwidth problems, while some rely instead on resilient audio codecs and others rely only on the robustness of transport protocols like User Datagram Protocol (UDP). These applications provide adequate quality of service (QoS) at the time being. However, with the rapid growth of Internet traffic, this QoS might degrade in the future. Peer to peer traffic (P2P) is growing steadily and is producing more traffic in the Internet than all other applications combined [19]. With the introduction of HDTV, we expect the traffic growth of P2P to continue in the future. The implication for the everyday users could be a phone that does not work. Most people will consider this very inconvenient. In the case of an emergency, it could be disastrous.

The questions we want to answer in this thesis are: What kind of traffic scenarios will disrupt a VoIP call? How badly congested is the network before serious disruption occurs? What modifications to end-point and network variables will improve the situation? In this thesis we will try to answer these questions.

1.3 Contributions

We have created simulations to investigate TCP's influence on interactive multimedia traffic. In our simulations we have been able to identify scenarios where this traffic suffers considerably due to TCP's influence. We have identified some of the changes in the network and in the end systems that make a difference in TCP's influence. Hopefully these findings will inspire more research in this field and contribute to the betterment of multimedia traffic applications.

1.4 Structure

This thesis is divided into five chapters. In chapter 2 we will go into the background for this thesis. We will describe the underlying technology which is the basis for VoIP as well as the network and data stream properties relevant to our simulation.

Chapter 3 will give an overview of how our simulation is designed and what aspect of the traffic we have considered interesting.

In chapter 4 we will present the results of our simulations together with thoughts and comments to the cause of these results.

Finally, in chapter 5 we will present our conclusions and possibilities for further research.

Chapter 2

Background

In this chapter, we give a short introduction to applications and protocols relevant to this thesis.

2.1 Transport Protocols

The two main transport protocols used in the Internet today is Transport Control Protocol (TCP) [20] and User Datagram Protocol (UDP) [23]. How these protocols behave and affect each other is a central part of this thesis, and in this section, we will give an overall review of their behaviour.

2.1.1 User Datagram Protocol

UDP is a connection-less transport protocol designed to enable data transfer with a minimum of protocol mechanisms. The header of a UDP packet contains little more than what is strictly needed to get a packet to its destination. The only error control is a checksum that can be used to verify that the packet has arrived correctly. Since UDP is connection-less, missing and faulty packets are ignored in silence. It is generally used in applications where speed is of more importance than correctness. It is often used in multimedia traffic, since a lot of multimedia traffic is time sensitive and resilient to moderate packet loss. UDP is often the preferred protocol for VoIP applications.

2.1.2 Transport Control Protocol

TCP is a connection oriented transport protocol. As opposed to UDP, it offers reliable data transfer. This is done by initiating a connection before transfer of data occurs. The connection serves to synchronize the starting sequence number of packets. This is done with a SYN packet. When data is transferred, the receiver can ask for retransmissions of lost or damaged packets as well as inform the sender if it is ready to receive more packets or not. All received data is confirmed by an acknowledgement (ACK) sent from the receiver to the sender. This ACK contains the sequence number of the last received byte. If an ACK is not received within a given time period, the transmitter will resend the corresponding data due to a timeout.

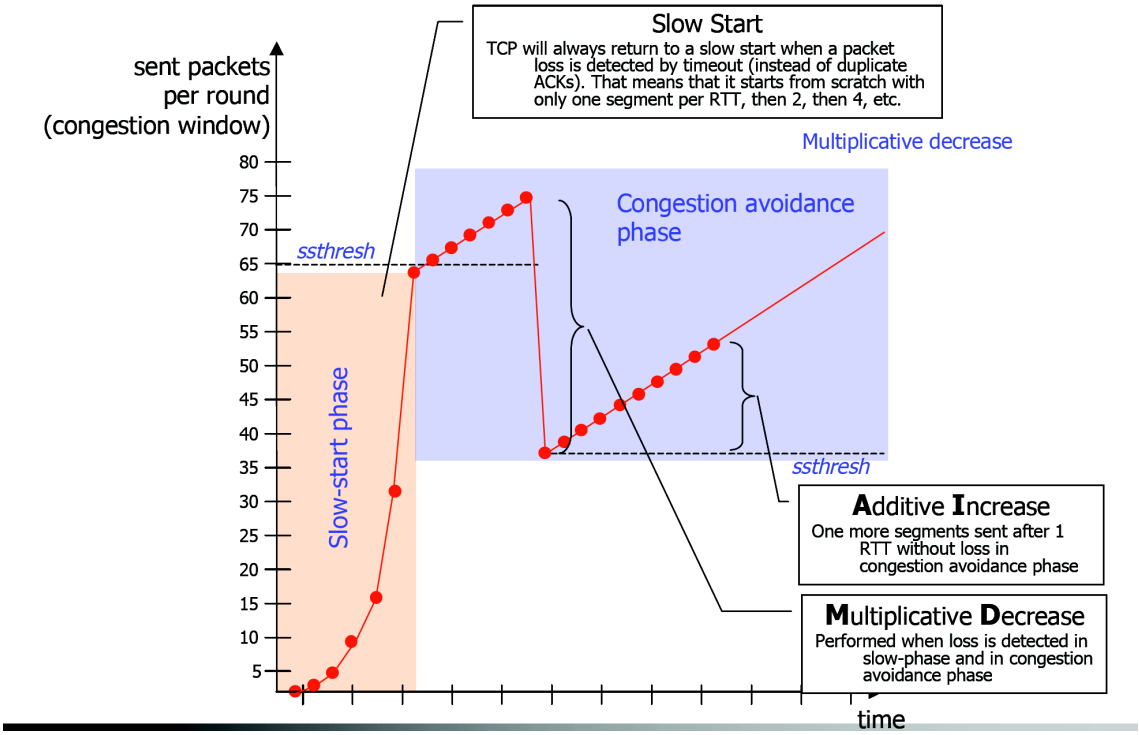

Figure 2.1: Illustration of TCPs congestion control [15].

Congestion Control

The fact that TCP has a connection and a concept of whether or not a packet is received is the basis for TCP congestion control [34]. TCP uses a congestion window to determine how many packets can be sent at any time before waiting to receive an

ACK. Normally an ACK will be received at the and of a round-trip-time (RTT). When TCP starts to transmit data the congestion window is set to one maximum segment size (MSS). TCP enters a phase called slow start where this congestion window is increased with one MSS for each ACK it receives until a slow start threshold is reached. When the slow start threshold is reached, TCP starts what is known as congestion avoidance where it increases the congestion window with one MSS for each RTT. If the receiver receives a packet out of order, it assumes this packet is lost and signals this to the sender by sending a duplicate of the last sent ACK. When such a duplicate ACK is received by the sender, the slow start threshold and the congestion window is set to half the current congestion window. TCP then resumes the congestion avoidance phase. This can be seen in figure 2.1. If a packet loss is discovered by timeout, the slow start threshold is set to half the current congestion window and the congestion window is set to one MSS. TCP then enters a new period of slow start. The TCP congestion control described here is known as TCP Reno. There are other kinds with slight variations to what is described here.

2.2 Interactive Multimedia Applications

Multimedia is defined as the combined use of several media, as sound and full-motion video, in computer applications [7]. This covers everything from VoIP to video streaming and multi-player online games. Interactive means acting one upon or with the other [7] and is used to describe the traffic generated between one or more users and / or a system. Most multimedia traffic in the Internet implies interactiveness to smaller or larger degree and this interactiveness affects the traffic patterns.

2.2.1 VoIP

Voice over IP is the encoding and packetization of voice and / or video and the subsequent transfer of this over the Internet. This enables users to use computers equipped with a speaker, microphone and an Internet connection as an alternative to the circuit switched phone system. It has become widely and wildly popular. In fact, Internet ser-

vice providers (ISP), like Telenor AS, have successfully advised customers to replace or compliment their legacy circuit switched phone line with a VoIP based system. Since its introduction in the mid 90's, VoIP has grown to be a billion dollar industry taking larger market shares from the legacy phone network every year [31]. AS more and more people and businesses depended on this technology it sets high demands on quality and reliability. In this thesis we have focused on voice traffic as generated by VoIP. Although pure voice traffic is only a small part of the variety of multimedia traffic in the Internet today, the characteristics of VoIP traffic is present in most of the other forms of multimedia traffic.

VoIP uses, as the name implies, IP packets to place and maintain the calls. Since IP inherently gives no guarantees the quality and reliability of Internet telephony are subject to the moment to moment transitions of the traffic load in the various parts of the Internet. Much has been done to overcome this weakness, however most of the solutions used today involve resilient audio codecs.

The main transport protocol used for VoIP is UDP. VoIPs traffic pattern generally consists of small packets, 20 to 160 bytes payload with 30 to 50 packets per second. The UDP protocol, being connection-less in nature, does not adapt to bandwidth problems. When the UDP protocol encounters bandwidth problems, packets are dropped by the intermediate systems if there are not enough resources left. Applications are expected to silently ignore lost packets preferably in a graceful manner unnoticed by the user. Table 2.1 shows a selection of VoIP codecs with their corresponding payload size and bit rate. Bear in mind that many of these protocols have variations with alternate payload sizes and Packets per second. There is a big difference in how well these codecs perform when they experience packet loss. Users will notice packet loss in ITU G.711 at 1 percent packet loss, while iLBC reports graceful degradation of quality as far as 30 percent packet loss. ITU-T has defined a set of latency thresholds for VoIP traffic. Users begin to get dissatisfied if the one-way delay exceeds 150-200 ms and that the maximum delay should not exceed 400 ms [18].

Codec	Bit Rate (kbps)	Payload Size (Bytes)	Packets per second
ITU G.711	64	160	50
ITU G.729	8	20	50
ITU G.723.1	6.3	24	34
ITU G.723.1	5.3	20	34
ITU G.726	32	80	50
ITU G.726	24	60	50
ITU G.728	16	60	34
iLBC	13.33	50	33.33
iLBC	15.20	38	50

Table 2.1: Examples of VoIP codecs with bandwidth consumption, payload size and packets per second [27] [17].

Most VoIP applications use a signalling protocol called Session Initiation protocol (SIP) [21] to create, modify and terminate sessions. It is also used to locate users and carry session descriptions to allow participants to agree on a set of compatible media types. The Real Time Protocol (RTP) [25] and its sister protocol RTP Control Protocol (RTCP) [25] are often used by VoIP applications to transport real-time data. Neither of these protocols are part of our simulations since they address other parts of VoIP traffic than we are focusing on in this thesis. Specially interested readers are referred to [21] and [25].

Silence suppression [30] is a technique to save bandwidth in VoIP calls. During periods of silence the application simply does not transmit packet. Since there is no speech there is no relevant data to transmit. This is of course an efficient way to save bandwidth.

2.2.2 Multi-player Online Games

Multi-player online games refer to the vast number of games that allow people to share a gaming experience through an Internet connection. These games include role-

playing games, real-time strategy games and first-person shooter games. There are several different games in each category, and they all have to exchange data with real-time demands to guarantee a good gaming experience for the player.

The traffic generated by multi-player online games has a lot in common with VoIP. In a typical first-person shooter game each player sends about 25 packet per second for a total of 15.7 kbps [13]. The traffic for the server is proportional to the number of clients with an average of 16 packets per second and 16.4 kbps per client connected. When there is a pause in the game, all traffic is suspended. This has similarities with silence suppression in VoIP, but it does not occur as frequently. The latency demands for online games vary greatly. Users start getting annoyed at 100 ms for first-person shooters, 500 ms for role-playing games and 1000 ms for real-time strategy games.

The results of this thesis may be somewhat applicable to this kind of traffic, however this will not be a direct part of our simulations.

2.2.3 Video on Demand

Video on demand (VoD) is the streaming of video at a user's request. VoD presents many of the same challenges as VoIP and some others as well. Although the main kind of traffic is similar to the one used in VoIP, VoD users expect the same kind of flexibility they get from their living room DVD player. This includes skipping forward or backward, pausing, resuming and more. While there is more data to be transferred the time restriction are not as stringent as in VoIP since it is easier to buffer data.

2.3 Other Applications

Since our thesis is a simulation of TCP's influence on multimedia traffic, a short description of some applications that utilize TCP is in order.

2.3.1 File Transfer Protocol

File transfer protocol (FTP) [24] is designed to enable users to access remote file storage systems and transfer data reliably and efficiently. Depending on the amount of

data to be transferred, FTP may consume all available bandwidth. FTP uses TCP as its transport protocol and will keep increasing its data-rate according to TCP congestion control algorithm. This enables FTP to transfer as much data as the network can handle. Because of TCP congestion control it will halve its transmission rate if a packet is lost.

2.3.2 World Wide Web

The World Wide Web is the total of all the web pages stored on web servers in the world. Its main basis is the Hypertext Transfer Protocol (HTTP) [22]. HTTP is an application level protocol originally designed to manage requests and transfers of Hypertext Markup Language (HTML) [11]. What makes the World Wide Web interesting for our thesis is the fact that most HTTP requests sent to web pages around the world are relatively small in size. This seriously affects the behaviour of the TCP congestion control since most of the requests will be completed before the TCP timeout triggers a reduction of the congestion window. This makes web traffic a lot more aggressive than other forms of traffic like FTP.

2.3.3 BitTorrent

BitTorrent [33] is a P2P protocol with millions of users world wide. It is designed to be a fast an efficient way to distribute files to many users. The general idea is that files are split into many equal sized blocks, which are usually 32-256 kB. Users can then download these blocks from multiple peers concurrently. This is done by sending a request to a peer in possession of a block and requesting a transfer of the block. Due to the size of the blocks TCP congestion control will have little time to react. This makes BitTorrent traffic as aggressive as web traffic.

2.4 Summary

In this chapter we have presented the two main transport protocols used in the Internet today. We have briefly described the main difference between them. The applica-

tions relevant to our thesis have been introduced and we have described some of their characteristics. We have seen that the demands on interactive multimedia traffic are stringent, and the network and applications intended to support it must have equally stringent demands.

To further investigate how these applications and protocols affect each other, we need testing. In the next chapter we will look at how our simulation was designed.

Chapter 3

Design

In this chapter we will present an outline of the simulation setup.We will look at what choices we have made and what we have chosen as a focus for the simulations.

3.1 Simulator Requirements

Designing a simulation required an understanding of what characteristics we needed to simulate. We wanted the possibility to simulate several different traffic types, both UDP based and TCP based. More than this, we wanted to be able to change network characteristics to measure the influence this would have on VoIP call. To be able to choose an adequate simulator we had to define what options we needed.

3.1.1 Traffic Types

First and foremost the simulator would need to be able to generate VoIP-similar traffic and TCP cross traffic at various rates and packet sizes. This is of course something most network simulators include. However with the simulations we had in mind, we needed the ability to simulate both VoIP with UDP transport as well as VoIP with TCP transport. We would also need to generate some TCP cross traffic for the VoIP calls. Web traffic generation was of the highest importance since this kind of traffic is more likely to pose serious interference with a VoIP call as discussed in 2.3.

3.1.2 Network Options

No network simulator would be any good unless we could thoroughly manipulate the aspects of the network. We needed to be able to model different kinds of networks to see what would change TCP's influence on the multimedia traffic. We wanted the ability to change queueing method and queue length as well as the latency in the network. The network bandwidth would also have to be changed if we were to understand the effects the network capacity had on our results and to show that our results are valid for higher bandwidths.

3.2 Network Simulator 2

We considered both JavaSim [4] and Network Simulator 2 (NS-2) [6] for our simulations. In the end we chose NS-2 because of its lower execution time, its wide array of possibilities and its reputation for consistent results. Having its roots as far back as 1989, the NS-2 simulator is constantly upgraded and improved. When we started our simulations the latest version was 2.30. For consistency, all our simulations have been run with this version although version 2.32 has been released by the time of this writing.

NS-2 is an open source community developed software and this is in no small part the reason that it is so extensive. Users can make their own extensions to the simulator, should it be needed, and contribute these to later versions of the simulator. The NS-2 manual thoroughly documents all the features of the program and is constantly updated. We use but a fraction of these features. We will describe the features that are most important to our simulations.

The interface of NS-2 is an object-oriented version of the Tool Command Language called OTcl, while the simulator itself is programmed in C++. The OTcl language is not considered to be a difficult language to learn. One of the main advantages to using a scripting language to make network simulations is that it allows the user to design simulations with an amount of freedom which no graphical user interface would allow.

NS-2 also comes with a network animator tool called Nam [5] which allows users to view their network layout and packet traversal through it.

3.2.1 Nodes and Links

Nodes and links are what make up a network in NS-2. The nodes correspond to real world machines, acting as end points if only one link is connected and as routers if more than one link is connected. The links correspond to cables. This is naturally the first step in creating a simulation, the simplest form of which is two nodes bound together by a link. When creating a link between two nodes we can choose between a simplex link or duplex link. When creating a link we also set bandwidth, link delay and queueing method. We can then assign a queue limit to the link.

3.2.2 Transport Protocols in an NS-2 Simulation

Transport protocols in NS-2 are modelled by transport agents. These correspond to real world transport protocols and have to be attached to the nodes before anything can be sent or received from them. The transport agent that interested us were of course the UDP agent and the TCP agent which represent UDP and TCP in the real world.

The UDP Agent

A UDP flow is simulated by creating a UDP agent and attaching it to a node. A null agent must also be created and attached to another node. Then the UDP agent is connected to the sink. When a traffic generator is attached to the UDP agent the traffic will flow from the corresponding node to the node with the attached null agent. UDP agents like UDP itself are simple and there is only one implementation for UDP agents in NS-2.

The TCP Agents

TCP traffic is simulated by TCP agents which come in a wide selection divided into two main categories: One-way TCP and FullTCP.

The first of these categories is, as the name says, a one-way sender and data flows only one way. When created, this agent is attached to a node. Although it is called a one-way TCP, we need an agent to respond with ACKs. Therefore we have to create a TCP sink agent, and attach it to another node. The TCP sender agent is connected to the TCP sink agent, and traffic will flow from the TCP sender agent to the sink agent, while ACKs flow the other way. The one-way TCP sender was the first to be implemented in NS-2, and consequentially, it has the widest range of congestion control options, while the newer FullTCP is only implemented with Reno congestion control.

The FullTCP can serve as both source and receiver of data. Here we create two TCP agents, and attach them to a node each. Then we connect the two agents, and instruct one of them to listen for an incoming connection, while the other will be the sender of the data. In our simulations we have used FullTCP, since the one-way TCP agent seems to pad its payload up to max payload size, no matter what the actual payload is. This resulted in TCP packets with 50 bytes payload having a total size of 1500 bytes, making any attempts at simulating CBR over TCP futile.

3.2.3 Application-Level Traffic in NS-2

The traffic that concerned us were web traffic, VoIP traffic and FTP traffic. In NS-2 these are simulated by the constant bit rate (CBR) traffic generator, the FTP application and the PackMime-HTTP traffic generator.

The CBR traffic generator is attached to an agent and produces packets at a constant rate. It must be parametrized with packet size and sending rate. It also has the option of randomness in the departure times of packets. With these options, we consider this to be a sufficient basis for simulating VoIP.

The FTP application simulates an FTP server, and will generate packets of a size as defined by the max TCP packet size defined in the TCP agents. It assumes there is always more data to send, and is only hindered by network bandwidth and TCP congestion control.

The PackMime-HTTP traffic generator generates web traffic based on the PackMime Internet traffic model [26], which is based on recent Internet traffic traces. Each instance of the traffic generator controls two applications: A PackMime-HTTP server application and a PackMime-HTTP client application. Each of these is attached to a FullTCP agent, and each of them can simulate hundreds of HTTP servers and HTTP clients. Since it is based on actual traces, the only variable is the rate at which new connections are started per second. This rate is the basis for the random number generator (RNG) making new connections. There are two more RNGs in the PackMimeHTTP, one for request-size and one for response-size.

3.2.4 Tracing and Analysis

To understand the influence of TCP on VoIP calls, we needed to know which VoIP packets were dropped and which VoIP packets arrived safely. We needed to know when the packets arrived to calculate jitter. For the cross traffic we needed to know the sequence number of the TCP packets to calculate the throughput of the cross traffic. Knowing where in the simulation events occured was also important. Lastly we needed to be able to distinguish the traffic types from each other.

NS-2 allows tracing of its simulations at any and all places in the simulation network. The simulator can be instructed to trace all packet data at all places in the network, or just at a specific link. The traces from the simulation is then written to a log file.

```
r 11.050447 0 1 ack 1500 ------- 272 10.272 11.272 78841 45087
r 11.051647 0 1 ack 1500 ------- 272 10.272 11.272 80301 45090
r 11.051679 0 1 ack 40 ------- 266 8.266 9.266 0 45096
d 11.05254 0 1 cbr 50 ------- 2 2.0 3.0 1 45915
r 11.052879 0 1 ack 1500 ------- 286 10.286 11.286 1 45091
d 11.053948 0 1 ack 385 ---A--- 259 18.259 19.259 1 45921
r 11.054079 0 1 ack 1500 ------- 316 4.316 5.316 1 45093
```

Figure 3.1: Excerpt from an NS-2 trace file.

An excerpt form an NS-2 trace file can be seen in figure 3.1. The first field in this excerpt indicates the kind of event that has been recorded. Here we see five receive events indicated by *r* and two drop events indicated by *d*. The next field is the time of the event measured to the microsecond. This is measured since the start of the simulation and is referred to as the time index. Following this we have two numbers that refer to nodes in the network. They indicate that these events happened at the link between node *0* and node *1*. Next is packet type and size. Then follows a series of flags where the *A* means congestion window reduced. The next field is a flow identifier followed by two fields indicating source and destination address. Then there is the sequence number and finally a unique packet identifier.

The fields that interested us were the recieve / drop field, the time index, the nodes describing which link it was, the packet type, packet size and the sequence number. With this information we would be able to make some conclusions to our simulations.

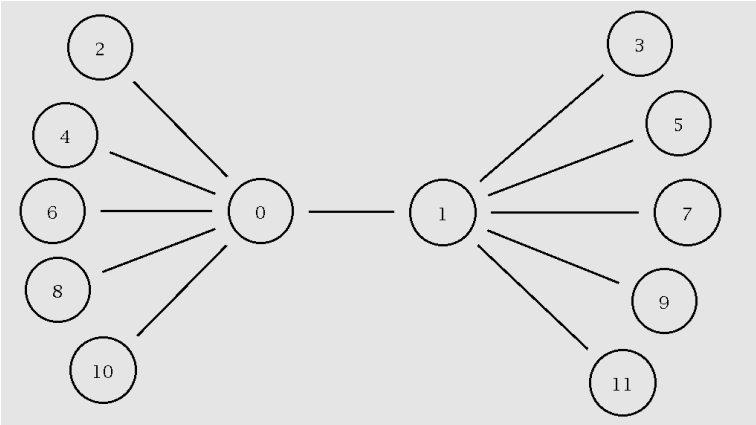

Figure 3.2: General network topology

3.3 Our Simulations

Since we were planning to investigate TCP's influence on VoIP traffic in our simulations, we had to create a network where there would be contention between these traffic types. To create real contention we needed a bottleneck that all the traffic would have to go through. We sent all traffic in the same direction intersecting in one node and consequently intersecting in the same queue before going through a single link to

the next node. This single link would then be our bottleneck. All traffic sources had the same bandwidth in their links as the bandwidth in the bottleneck.

The resulting topology of the experiments became generalized dumbbell network as seen in figure 3.2. Even numbered nodes represent data sources and odd numbered nodes represent data receivers. All sources and receivers are implemented as pairs with sources numbered as *n* and receivers *n+1* except for node *0* and node *1* which represent the bottleneck. Each of the node pairs can be configured as either a PackMime-HTTP application, a CBR-sender and receiver using UDP or TCP or an FTP-server and client. The OTcl script that we use parses the arguments given on the command line and most of the parameters in the simulation can be modified this way. All our trace data was gathered at the bottleneck between node *0* and node *1*.

The PackMime-HTTPs are all scheduled to start at 0 seconds from simulation start. The FTP-application increases its transmission rate according to TCP congestion control. To get variations in the cross-traffic generated by FTP-applications all of these applications have a random start time between 0 and 3 seconds from simulation start. 10 seconds after simulation start TCP based CBR begins its transmissions and at 11 seconds after simulation start the UDP based CBR traffic starts. This gives the cross-traffic about 10 seconds of warm-up time. The simulation ends after 31 seconds, which gives us 21 seconds of trace data for the TCP based CBR and 20 seconds of trace data for the UDP based CBR. The source code for our simulations can be found in appendix A.

3.4 Summary

In this section we have presented the network simulator used in our simulations. We have explained which options in the network simulator interests us and how we were going to collect the relevant data from our simulations. Finally we explained the general topology of our simulations. In the next chapter we will present the values of the variables we chose for our simulation and the results we got.

Chapter 4

Simulations and Results

In this chapter, we will present an overview of the various simulations we have run, followed by the results themselves.

4.1 Simulation Overview

Our default simulation is a network as described in 3.3 with 10 PackMime-HTTP applications acting as cross traffic for one UDP-based CBR traffic generator which represents our VoIP call. All links in the network have the same bandwidth which we have chosen to be 10 Mbps. The number of PackMime-HTTP applications was chosen after a few initial simulations which determined that this was a reasonable number to create a serious bottleneck, since lower numbers of PackMime-HTTP applications produced less contention. There is a wide range of VoIP codecs, each with their own combination of bandwidth and payload. We have chosen to try and make an amalgam of the codecs shown in table 2.1. We chose to use 50 bytes packet size with a rate of 32 kbps for our CBR traffic. As a result of this, CBR produces one packet every 12.5 ms. The end-to-end delay in the network is 120 ms, which is approximately the delay Oslo, Norway to New York City, US [1]. This delay is further split into three equal parts: 40 ms delay before the bottleneck, 40 ms delay in the bottleneck and 40 ms delay after the bottleneck. The queueing method used in the network is tail-drop. When choosing a queue length we used what is now a considered a rule-of-thumb introduced in [35],

[1] Obtained with traceroute to www.nytimes.com

which states that the buffer in routers should be the product of the bandwidth and the average RTT of flows going through the router. In our case, this calculates to 300KB and resulted in a queue length of 200 packets since our maximum transfer unit (MTU) in the network is 1500 bytes. The MTU was chosen at 1500 because this is the MTU of the Ethernet standard [1]. The rate of the HTTP connections was incremented in steps of 100 until significant packet loss occurred and beyond. All simulations were run 50 times to get a statistical base, with the exception of the 100 Mbps network simulations. All plots are made with the mean of our data. The 100 Mbps network simulations had to be reduced to twenty runs due to the precessing time of these simulations. All data is based on events after the warm-up phase which ends at time index 10 seconds into the simulation. The simulation ends after 31 seconds.

From this default simulation, we created new simulations where one or more variables were changed in each. We experimented with changing a few endpoint variables in our simulation. Keeping the packet size constant, we tried two identical simulations except that a CBR rate of 13 kbps (one packet every 30.77 ms) was chosen in one simulation and a 64 kbps (one packet every 6.25 ms) in the other. After these two experiments, we tried to simulate silence suppression by stopping the CBR traffic at certain intervals while changing none of the settings from the default simulation. We then ran a simulation where we changed the transport protocol from UDP to TCP. We also ran some experiments where we substituted the cross traffic generated by the PackMime-HTTP applications with FTP applications. Furthermore, the network itself presents a whole other set of variables to be modified. First off, we tried to increase the queue length in the network from the default 200 packets to 1000 in one simulation and 2000 in another. After this, we tried changing the distance to the bottleneck, while keeping the end-to-end delay unchanged. We also ran a simulation where we changed the queueing method from tail-drop to Random Early Detection (RED) [28]. Finally, we considered a scalability test was in order. To do this, we increased the bandwidth in the network to 100 Mbps, queue length to 2000 packets and number of PackMime-HTTP applications to 100.

Having decided what to do there was the question of processing power. While NS-2 is an excellent simulator, the processing needs for the number of simulations we needed to run to get a decent statistical significance were quite large. Fortunately, we had access to a small cluster of computers configured with Condor [32], which allowed us to submit our simulations in bulk for distributed processing.

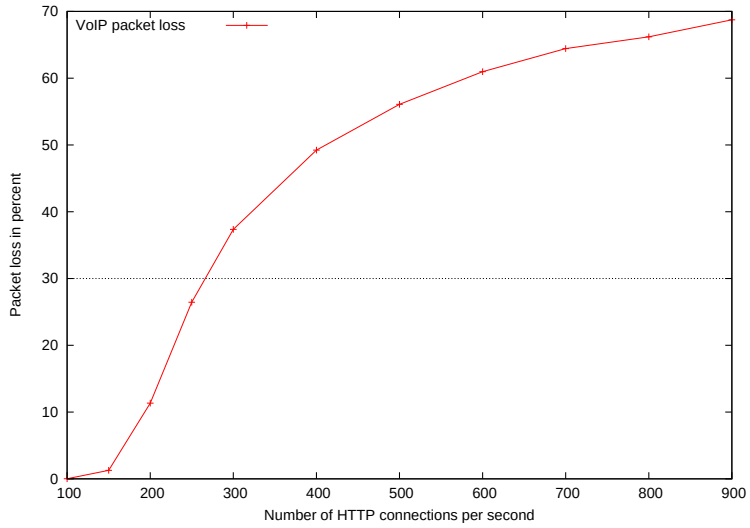

Figure 4.1: Packet loss plot for 32 kbps VoIP call with HTTP cross traffic in a 10 Mbps network with queuing method tail-drop and a queue length 200 packets.

4.2 Default Simulation

The first of these simulations was to identify the amount of traffic necessary to create significant packet loss for a single VoIP call. Figure 4.1 shows a plot of the packet loss in relation to the number of HTTP connections per second. Given that the more resilient codecs like iLBC [37] report graceful degradation of voice quality until the packet loss exceeds 30 percent, this was what we considered the threshold value. As we can see by the plot, this value is exceeded at just above 250 HTTP connections per second in the default simulation scenario. Table 4.1 shows the statistical data from this simulation. In table 4.2, the retransmission statistics for the HTTP connections are shown. It is no surprise that the TCP retransmissions increase rapidly, nearly tripling, from 200 connections per second to 250 connections per second as we see the same

Connections per second (HTTP)	Average packet loss per call (percent)	Standard deviation packet loss per call	Median packet loss per call	Minimum packet loss per call	Maximum packet loss per call	Minimum jitter (ms)	Maximum jitter (ms)	95th percentile minimum jitter (ms)	95th percentile maximum jitter (ms)
100	0.02	0.08	0.00	0.00	0.50	-12.460	22.135	-7.168	7.184
150	1.28	2.47	0.31	0.00	13.12	-12.460	24.952	-9.288	9.369
200	11.33	7.74	10.37	0.25	31.42	-12.460	26.613	-8.668	9.531
250	26.45	7.66	28.57	4.85	39.94	-12.428	21.236	-7.021	8.495
300	37.36	4.76	38.23	26.23	45.00	-11.832	21.226	-6.094	7.684
400	49.23	2.59	49.51	42.94	53.74	-10.410	20.518	-5.715	7.366
500	56.08	2.28	56.24	49.84	60.78	-11.000	21.909	-5.606	7.050
600	60.97	1.76	61.11	57.94	65.87	-11.860	18.170	-5.461	6.889
700	64.44	1.80	64.17	61.12	68.53	-11.425	18.112	-5.506	6.654
800	66.20	1.45	66.01	62.82	68.82	-10.485	20.654	-5.350	6.686
900	68.74	1.58	68.81	64.45	71.91	-10.702	20.496	-5.372	6.588

Table 4.1: Packet statistics for a 32 kbps VoIP call with bottleneck placed equidistantly from sender and receiver in a network with 10 Mbps bandwidth, queueing method tail-drop and 200 packets queue length.

Connections per second(HTTP)	Average retransmissions per packet	Standard Deviation Retransmissions	Median retransmissions	Minimum retransmissions	Maximum retransmissions
100	0.00	0.00	0.00	0.00	0.01
150	0.01	0.03	0.00	0.00	0.15
200	0.13	0.10	0.11	0.00	0.42
250	0.35	0.12	0.40	0.05	0.58
300	0.57	0.11	0.57	0.34	0.79
400	0.92	0.08	0.94	0.73	1.06
500	1.22	0.09	1.22	0.98	1.37
600	1.51	0.07	1.50	1.38	1.66
700	1.76	0.09	1.75	1.56	1.92
800	1.95	0.07	1.94	1.80	2.13
900	2.17	0.08	2.16	1.93	2.39

Table 4.2: Statistical data for TCP retransmissions in a network with 10 Mbps bandwidth, queueing method tail-drop and 200 packets queue length.

growth in packet loss for the VoIP call. In a 10 Mbps network, 250 HTTP connections per second is not unlikely, and the same packet drop, up to 30 percent, could occur in the Internet [29].

4.3 Endpoint Controlled Variables

In this section, we present the results we got from varying the endpoint controlled variables. Having established a basis for comparison, we could now experiment with some of the endpoint controlled variables to see whether these would affect the packet loss of the VoIP call.

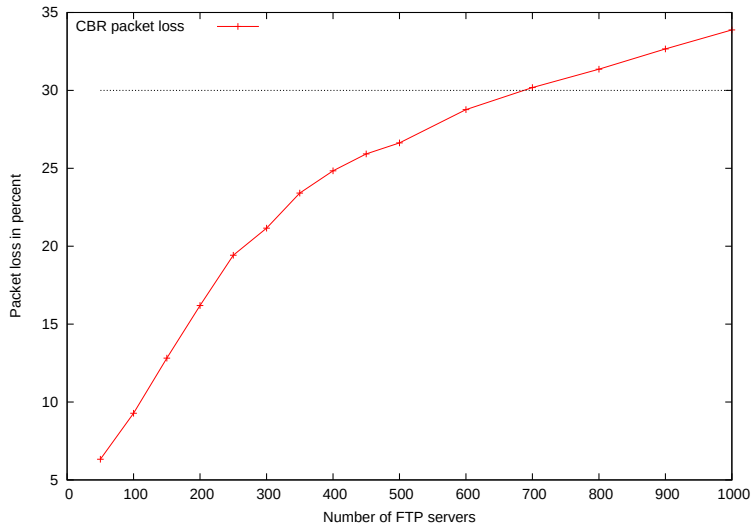

Figure 4.2: Packet loss plot for a 32 kbps VoIP call with cross traffic generated from FTP servers in a 10 Mbps network with queuing method tail-drop and a queue length 200 packets.

4.3.1 Changing the Cross Traffic to FTP

With the same network and VoIP specifications as in the default simulation, we changed the cross-traffic from HTTP to FTP applications. Figure 4.2 shows a plot of the packet loss experienced by the VoIP call in relation to the number of FTP servers. We see that the threshold, for 30 percent loss in the VoIP call, for the FTP cross traffic is at about 700 servers. The bandwidth used in this experiment is only 10 Mbps and we consider 700

Number of FTP servers	Average packet loss per call (percent)	Standard deviation packet loss per call	Median packet loss per call	Minimum packet loss per call	Maximum packet loss per call	Mini-mum jitter (ms)	Maxi-mum jitter (ms)	95th percentile minimum jitter (ms)	95th percentile maximum jitter (ms)
50	6.33	0.7	6.33	4.98	8.68	-12.460	17.540	-6.460	4.340
100	9.28	0.44	9.3	8.37	10.23	-12.460	17.540	-5.260	5.540
150	12.82	0.66	12.9	11.31	14.38	-12.460	16.340	-5.260	5.540
200	16.19	0.68	16.25	14.65	17.53	-11.260	13.940	-5.260	5.540
250	19.42	0.81	19.39	17.85	21.44	-11.260	15.140	-5.260	5.540
300	21.16	1.15	21.54	17.93	22.63	-11.260	16.404	-5.260	4.340
350	23.41	0.59	23.38	21.65	24.95	-11.260	17.636	-4.060	4.340
400	24.84	0.66	24.8	23.44	26.45	-10.060	14.004	-4.060	4.340
450	25.92	0.61	25.89	24.65	27.3	-11.228	16.372	-4.060	3.140
500	26.63	0.67	26.49	25.57	28.24	-10.060	12.868	-4.060	3.140
600	28.77	0.71	28.65	27.47	30.21	-10.060	12.740	-4.028	3.140
700	30.18	0.8	30.11	28.49	32.49	-8.860	7.940	-2.860	3.140
800	31.36	0.74	31.52	29.52	33.08	-8.860	10.340	-2.860	2.5200
900	32.66	0.9	32.6	30.76	34.97	-9.996	7.940	-2.860	2.520
1000	33.88	0.87	33.78	31.84	35.88	-8.860	10.340	-2.860	1.940

Table 4.3: Statistical data for a 32 kbps VoIP call with bottleneck placed equidistantly from sender and receiver in a network with 10 Mbps bandwidth, queueing method tail-drop and 200 packets queue length.

FTP-servers on such a small link unlikely. Statistical data for the VoIP traffic is shown i table 4.3. An interesting effect with the FTP cross-traffic is how small jitter experienced by the VoIP traffic becomes when the FTP traffic increases. This is because the number of FTP-servers competing with each other ensures that their TCP congestion windows remain at or just above the minimum, which reduces the burstiness of the traffic. This is also seen in table 4.4 which shows the retransmission data for the TCP packets generated by the FTP servers. When the number of FTP servers exceeds 800, about a third of the packets are retransmissions.

4.3.2 Varying the Bandwidth used by the VoIP Call

The next simulations were one with a VoIP call rate of 13 kbps and one with a VoIP Call rate of 64 kbps. Otherwise, the simulations were identical to our default simulation. In figure 4.3, we see the VoIP packet loss for these simulations compared to the VoIP

Number of FTP servers	Average retransmissions per packet	Standard Deviation Retransmissions	Median retrans-missions	Minimum retrans-missions	Maximum retrans-missions
50	0.02	0.00	0.02	0.02	0.02
100	0.05	0.00	0.05	0.05	0.06
150	0.09	0.00	0.09	0.08	0.10
200	0.14	0.00	0.14	0.13	0.14
250	0.18	0.00	0.18	0.17	0.19
300	0.22	0.02	0.22	0.18	0.24
350	0.26	0.01	0.26	0.25	0.28
400	0.30	0.01	0.30	0.28	0.31
450	0.33	0.01	0.33	0.31	0.34
500	0.36	0.01	0.36	0.34	0.37
600	0.42	0.01	0.42	0.40	0.43
700	0.47	0.01	0.47	0.45	0.48
800	0.51	0.01	0.51	0.48	0.53
900	0.55	0.01	0.55	0.54	0.58
1000	0.60	0.01	0.59	0.58	0.61

Table 4.4: Statistical data for FTP retransmissions in a network with 10 Mbps bandwidth, queueing method tail-drop and 200 packets queue length.

Connec-tions per second (HTTP)	Average packet loss per call (percent)	Standard deviation packet loss per call	Median packet loss per call	Minimum packet loss per call	Maximum packet loss per call	Mini-mum jitter (ms)	Maxi-mum jitter (ms)	95th percentile minimum jitter (ms)	95th percentile maximum jitter (ms)
100	0.02	0.11	0.00	0.00	0.77	-27.936	31.862	-12.547	12.018
150	0.55	1.00	0.15	0.00	5.60	-29.505	37.339	-16.335	15.856
200	9.26	6.04	9.01	0.15	23.68	-29.092	37.462	-15.508	16.461
250	26.66	6.94	28.43	9.44	44.51	-25.103	34.420	-11.575	13.336
300	35.98	4.21	35.63	26.01	44.96	-19.850	27.890	-10.140	12.035
400	47.78	2.82	48.07	41.33	54.11	-18.256	31.791	-9.640	11.283
500	55.04	2.10	54.83	51.39	59.51	-17.555	28.751	-9.363	11.062
600	59.92	1.86	60.09	54.94	63.83	-18.496	22.613	-9.022	10.506
700	62.96	2.31	62.78	59.01	68.67	-17.541	27.051	-8.863	10.337
800	65.14	2.10	64.81	61.73	70.32	-17.313	32.448	-8.938	10.186
900	67.66	2.34	67.49	63.12	72.53	-18.239	22.393	-8.682	10.121

Table 4.5: Statistical data for a 13 kbps VoIP call with bottleneck placed equidistantly from sender and receiver in a network with 10 Mbps bandwidth, queueing method tail-drop and 200 packets queue length.

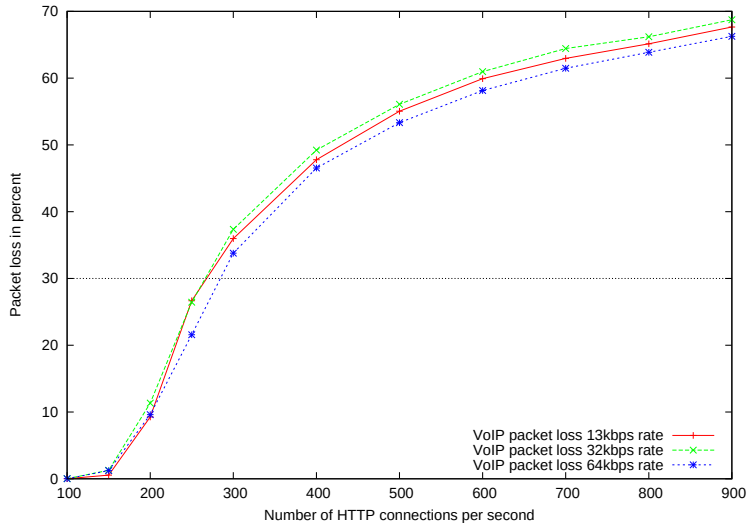

Figure 4.3: Comparison between packet loss for three different VoIP calls with different bandwidth requirements.

packet loss in our default simulation. Each call is placed alone in different simulations subjected to the same cross traffic. The plot indicates small differences between the packet loss of the different VoIP call rates. The 32 kbps VoIP call seems to have the greatest packet loss, the 64 kbps VoIP call has the lowest packet loss and the 13 kbps is in the middle between the two others. We see no clear pattern to explain this, but it could be a statistical anomaly. The statistical data for this plot is shown in tables 4.1, 4.5 and 4.6. If we look at the jitter experienced by these calls, we see that the jitter significantly decreases with higher VoIP call rate. This is, of course, a consequence of using the same packet size for all the VoIP calls, which means the average time difference between packets gets smaller when the rate is increased. Our network layout also precludes the reordering of packets, which means the negative jitter can not go below the time difference between packet transmissions.

4.3.3 Silence Suppression

A lot of VoIP systems today use silence suppression to save bandwidth. This made us think it was prudent to investigate if this had any effect on the percentage of packets dropped. We compared two streams, one with and one without silence suppression. Both VoIP calls use a bandwidth of 32 kbps, are run in individual simulations and are

Connections per second (HTTP)	Average packet loss per call (percent)	Standard deviation packet loss per call	Median packet loss per call	Minimum packet loss per call	Maximum packet loss per call	Minimum jitter (ms)	Maximum jitter (ms)	95th percentile minimum jitter (ms)	95th percentile maximum jitter (ms)
100	0.02	0.07	0.00	0.00	0.38	-6.210	13.663	-4.978	4.686
150	1.22	2.13	0.46	0.00	10.62	-6.210	18.918	-5.670	5.982
200	9.61	7.07	7.10	0.38	26.66	-6.210	17.918	-5.263	6.130
250	21.58	6.14	23.24	3.97	31.60	-6.210	16.000	-4.753	5.886
300	33.79	3.70	34.37	24.53	43.61	-6.158	15.960	-4.125	5.507
400	46.54	2.49	46.60	38.40	51.66	-6.146	14.910	-3.860	5.380
500	53.34	1.89	52.91	49.61	58.24	-6.082	13.676	-3.737	5.337
600	58.14	1.49	58.20	54.81	61.50	-6.082	14.510	-3.700	5.129
700	61.48	1.20	61.61	58.68	64.29	-6.082	14.256	-3.703	5.090
800	63.87	1.23	63.93	60.95	66.35	-6.082	16.725	-3.682	5.023
900	66.26	1.04	66.21	64.18	68.61	-6.082	16.075	-3.650	5.007

Table 4.6: Statistical data for a 64 kbps VoIP call traffic with bottleneck placed equidistantly from sender and receiver in a network with 10 Mbps bandwidth, queueing method tail-drop and 200 packets queue length.

Connections per second (HTTP)	Average packet loss per call (percent)	Standard deviation packet loss per call	Median packet loss per call	Minimum packet loss per call	Maximum packet loss per call	Minimum jitter (ms)	Maximum jitter (ms)	95th percentile minimum jitter (ms)	95th percentile maximum jitter (ms)
100	0.02	0.09	0.00	0.00	0.53	-12.460	20.295	-7.564	7.060
150	1.35	1.74	0.74	0.00	8.15	-12.428	23.572	-9.452	9.652
200	12.37	7.05	11.62	0.00	29.65	-12.396	22.876	-8.581	9.563
250	26.38	6.35	26.83	9.81	38.17	-12.300	18.294	-6.929	8.349
300	37.98	3.77	38.71	28.65	44.65	-12.268	19.795	-5.971	7.836
400	50.55	3.01	50.42	44.81	58.07	-11.448	20.606	-5.738	7.292
500	57.02	2.24	57.24	50.73	61.72	-10.543	18.359	-5.571	6.968
600	61.73	1.90	62.02	57.02	66.60	-12.014	19.180	-5.359	6.984
700	64.76	1.71	64.56	60.55	68.69	-11.309	16.589	-5.466	6.651
800	67.18	1.72	67.55	63.81	70.19	-10.625	19.773	-5.372	6.551
900	68.85	1.60	68.90	65.31	71.92	-11.597	19.405	-5.292	6.495

Table 4.7: Statistical data for a 32 kbps VoIP call using silence suppression and bottleneck placed equidistantly from sender and receiver in a network with 10 Mbps bandwidth, queueing method tail-drop and 200 packets queue length.

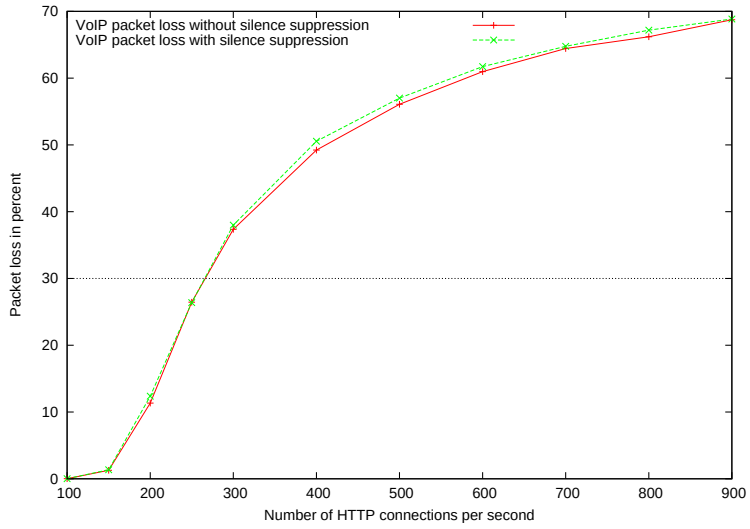

Figure 4.4: Comparison between packet loss for a 32 kbps VoIP call with and without silence suppression.

subjected to the same cross traffic. The simulation details are the same as in our default simulation, and the VoIP call without silence suppression is the same as in our default simulation. The silence suppressed VoIP call transmits packets for two seconds, pauses for two seconds, resumes for four seconds, pauses for six seconds and resumes for six seconds. As seen in figure 4.4, there is very little clear difference between the packet loss experienced by the two calls up to the threshold. The silence suppressed call is has a slightly higher packet loss. This is probably because, the cross traffic take up the available bandwidth left by the silenced VoIP call. This will result in additional packet loss for the VoIP call when it resumes its transmissions. Statistical data for these calls are shown in table 4.1 and table 4.7. As we can see in these tables, all the packet loss statistics show the same slight increase in packet loss. This indicates that silence suppression has a small disadvantage for VoIP communication, but there is of course an advantage to the network itself as this results in decreased load.

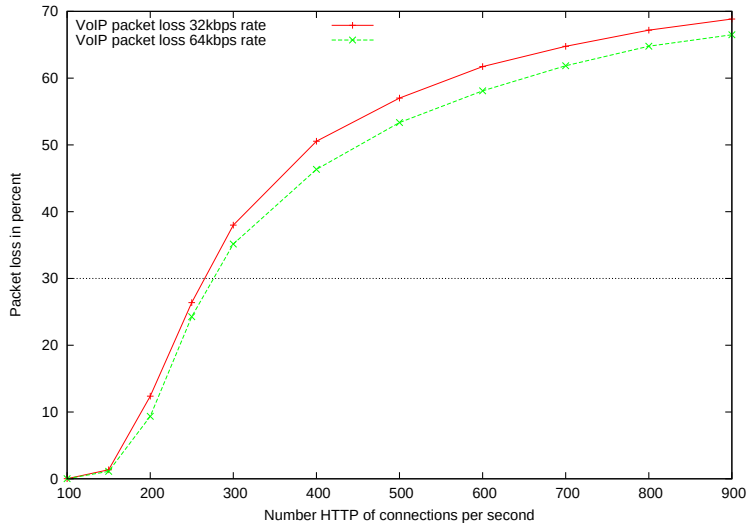

Figure 4.5: Comparison between silence suppressed 32 kbps and 64 kbps VoIP calls.

Connections per second (HTTP)	Average packet loss per call (percent)	Standard deviation packet loss per call	Median packet loss per call	Minimum packet loss per call	Maximum packet loss per call	Minimum jitter (ms)	Maximum jitter (ms)	95th percentile minimum jitter (ms)	95th percentile maximum jitter (ms)
100	0.02	0.09	0.00	0.00	0.63	-6.210	14.190	-4.978	4.671
150	1.11	2.07	0.10	0.00	8.83	-6.210	16.974	-5.611	5.903
200	9.36	6.73	8.15	0.21	25.30	-6.210	17.463	-5.324	6.180
250	24.28	6.92	24.93	4.60	38.53	-6.210	14.362	-4.673	5.854
300	35.15	4.80	35.69	20.69	42.96	-6.210	13.198	-4.151	5.618
400	46.34	2.84	46.84	39.72	51.36	-6.114	15.827	-3.850	5.329
500	53.35	2.05	53.40	48.45	58.36	-6.114	13.567	-3.820	5.336
600	58.10	1.70	58.34	54.12	61.12	-6.082	14.550	-3.763	5.173
700	61.85	1.44	61.84	59.00	65.13	-6.082	13.150	-3.722	5.161
800	64.78	1.08	64.58	61.32	67.28	-6.082	13.614	-3.657	5.053
900	66.50	1.29	66.75	62.65	68.89	-6.050	16.815	-3.714	4.893

Table 4.8: Statistical data for a 64 kbps VoIP call using silence suppression and bottleneck placed equidistantly from sender and receiver in a network with 10 Mbps bandwidth, queueing method tail-drop and 200 packets queue length.

4.3.4 Varying the Bandwidth used by the VoIP Call Using Silence Suppression

In fig. 4.5 we see the packet loss for two different VoIP calls. Both using silence suppression, one with a bandwidth of 32 kbps and one with a bandwidth of 64 kbps. Each

call is placed alone in different simulations subjected to the same cross traffic. The results indicate that the 64 kbps VoIP call performs better than the 32 kbps VoIP call, which are the same results we got in 4.3.2 when we did not use silence suppression. The data for this graph can be found in tables 4.7 and 4.8.

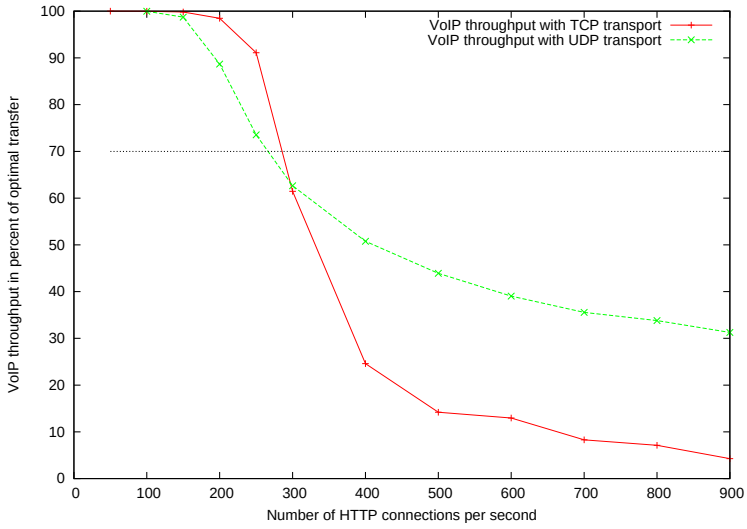

Figure 4.6: Comparison between packet loss for a VoIP call with UDP transport and VoIP call with TCP transport.

4.3.5 Alternative Transport

Although most VoIP applications use UDP as the main transport protocol there are networks which disallow UDP packets. Some VoIP programs such as Skype [10] have solved this by using the TCP transport protocol as a substitute in such networks. Figure 4.6 shows a comparison between a UDP-based VoIP call and a TCP-based VoIP call. The UDP-based VoIP call is the same as the one in our default experiment. Due to the fact that TCP does not loose packets, a packet loss plot would be meaningless in this case. Instead, we have plotted the amount of data that was recieved for VoIP calls in the duration of the simulation. Subsequently the threshold line is set at 70 percent which means that 30 percent of the data did not come through before the simulation ended. The TCP-based VoIP call retransmits lost packets. When such a retransmit is received at the destination, this id counted as part of the total throughput. This is the main reason TCP performs better up to the cross traffic level of 300 connections per

Connections per second (HTTP)	Average throughput per call in percent of optimum	Standard deviation through-put put	Median throughput per call	Minimum throughput per call	Maximum throughput per call
50	100.00	0.00	100.00	100	100
100	99.98	0.11	100.00	99.22	100
150	99.82	0.24	100.00	99.34	100
200	98.48	6.16	99.40	55.85	99.88
250	91.11	21.02	99.34	0.00	99.52
300	61.45	33.27	63.58	0.00	99.46
400	24.59	26.54	15.71	0.00	81.69
500	14.19	18.23	3.49	0.00	63.48
600	12.95	16.56	2.62	0.00	45.67
700	8.28	13.74	0.00	0.00	59.24
800	7.11	10.71	0.87	0.00	37.30
900	4.27	8.76	0.00	0.00	33.20

Table 4.9: Statistical data for a 32 kbps VoIP call using TCP transport with bottleneck placed equidistantly from sender and receiver in a network with 10 Mbps bandwidth, queueing method tail-drop and 200 packets queue length.

second. This must however be seen in relation with the fact that retransmission of packets in TCP will lead to increased jitter, and arrive too late to be useful. The data for the TCP-based VoIP call can be found in 4.9. The data used for the UDP-based call is the same as in table 4.1. These tables show that in the minimum per-call throughput, the TCP-based VoIP reaches 0 percent throughput at 250 HTTP connections per second. It is worth noting that the median throughput at this rate of cross traffic is 99.40, which would indicate that most of the TCP VoIP calls still perform well. We would like to emphasize that this test is not very fair towards TCP as a transport protocol. This is due to the fact that we have no setup phase for the VoIP traffic, but NS-2 will send SYN packets to initiate a TCP connection. This means that the TCP-based VoIP for all intents and purposes is subjected to a setup phase, which the UDP-based VoIP does not need. TCPs SYN packets have a long timeout. Due to the length of the simulation, this means that three consecutively lost SYN packets will result in a 100 percent packet loss for the VoIP call.

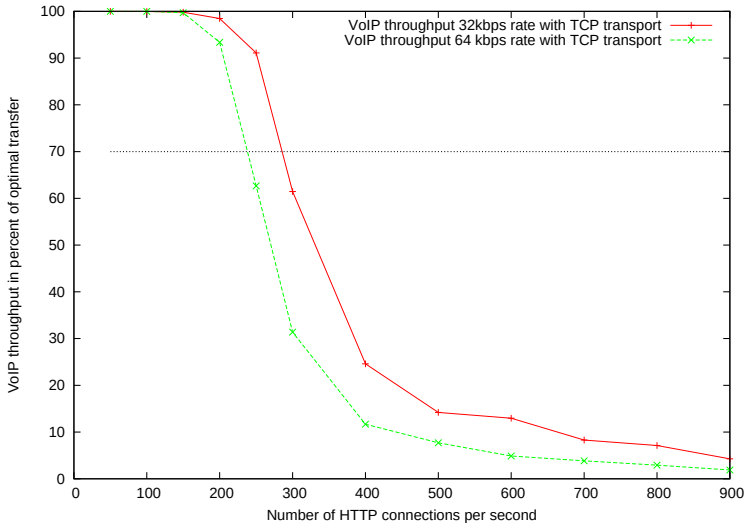

Figure 4.7: Comparison between 32 kbps and 64 kbps TCP-based VoIP calls in a 10 Mbps network.

4.3.6 Varying the Bandwidth of the VoIP Call with TCP-based VoIP

In this simulation we wanted to find out if varying the rate of TCP-based the VoIP call would have similar effect as varying the rate of the UDP-based VoIP call. The simulation was done in a simulation identical to the one in 4.3.5, but with a VoIP rate of 64 kbps. The comparison between this simulation an the one in 4.3.5, can be seen in figure 4.7. The statistical data for these calls can be seen in tables 4.9 and 4.10. We can clearly see that the 64 kbps VoIP call performs significantly worse than the 32 kbps VoIP call. This is because of TCP congestion control. When the number of packets increase, the probability for a lost packet increases and it takes only one lost packet for multiplicative decrease to take place. This tells us that for TCP-based VoIP, larger and fewer packets is a better alternative to many small packets. This is ,of course, only true as long as the packet size is small enough that least four packets are sent per RTT to give TCP congestion control the ability to use fast retransmit.

4.4 Network Variables

Having simulated the impact of end point administered variables, we proceeded to investigate how the network variables affect the packet loss percentage. These variables

Connections per second (HTTP)	Average throughput per call in percent of optimum	Standard deviation through-put put	Median throughput per call	Minimum throughput per call	Maximum throughput per call
50	100.00	0.00	100.00	100.00	100.00
100	99.99	0.06	100.00	99.55	100.00
150	99.70	0.34	99.81	98.39	100.00
200	93.37	14.17	99.34	35.75	100.00
250	62.67	29.47	57.88	4.36	99.46
300	31.39	16.42	33.42	0.00	62.95
400	11.68	14.38	6.97	0.00	56.55
500	7.71	10.01	3.05	0.00	32.48
600	4.88	7.44	1.71	0.00	25.31
700	3.84	6.60	0.87	0.00	26.08
800	2.91	4.01	1.69	0.00	17.63
900	1.88	3.81	0.00	0.00	19.27

Table 4.10: Statistical data for a 64 kbps TCP-based VoIP call with bottleneck placed equidistantly from sender and receiver in a network with 10 Mbps bandwidth, queueing method tail-drop and 200 packets queue length.

include queue length and queueing method at the routers, the distances in the network and of course bandwidth available in the network. Some of the changes made in the network itself yielded interesting results.

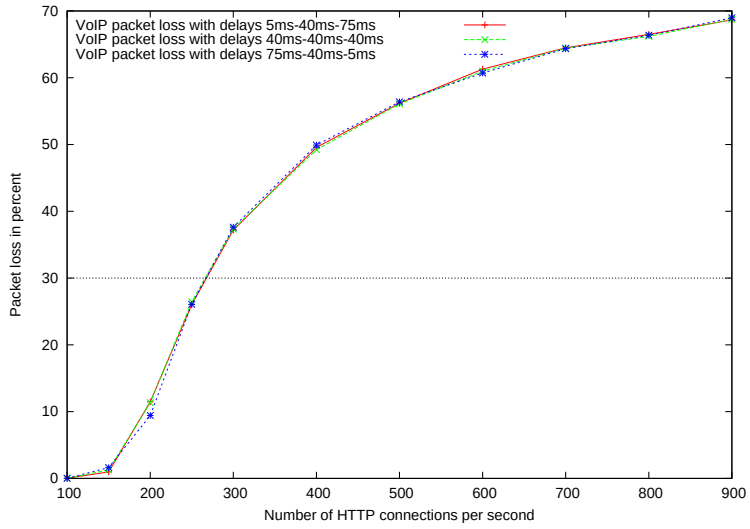

Figure 4.8: Packet loss for a 32 kbps VoIP call with HTTP cross traffic and varying distances to the bottleneck.

Connections per second (HTTP)	Average packet loss per call (percent)	Standard deviation packet loss per call	Median packet loss per call	Minimum packet loss per call	Maximum packet loss per call	Minimum jitter (ms)	Maximum jitter (ms)	95th percentile minimum jitter (ms)	95th percentile maximum jitter (ms)
100	0.03	0.12	0.00	0.00	0.69	-12.460	21.216	-7.564	7.044
150	0.98	2.06	0.28	0.00	11.35	-12.460	23.376	-9.295	9.376
200	11.47	7.96	11.96	0.00	34.84	-12.460	25.516	-8.605	9.609
250	26.03	7.80	28.36	7.43	38.49	-12.332	23.248	-6.927	8.424
300	37.19	4.60	37.23	26.08	46.24	-11.487	21.080	-6.056	7.864
400	49.57	2.40	49.95	45.02	55.21	-11.367	23.161	-5.688	7.362
500	56.18	1.53	56.19	53.17	59.18	-10.088	20.884	-5.602	7.096
600	61.29	1.48	61.43	57.59	63.83	-10.678	18.510	-5.461	6.868
700	64.48	1.72	64.42	61.40	68.71	-10.759	20.983	-5.436	6.659
800	66.50	1.42	66.64	63.73	69.40	-11.811	17.246	-5.302	6.622
900	68.66	1.35	68.59	65.62	71.50	-10.734	22.573	-5.366	6.376

Table 4.11: Statistical data for a 32 kbps VoIP call with bottleneck placed 5 ms from the sender and 75 ms from the receiver in a network with 10 Mbps bandwidth, queueing method tail-drop and 200 packets queue length.

4.4.1 Varying the Distance to the Bottleneck

This simulation was to determine if the placement of the bottleneck would have any significant effect on the packet loss. Figure 4.8 shows a comparison between a 32kb VoIP call in three different networks. One network is identical to our default simulation with 40 ms delay before the bottleneck and 40 ms delay after it. The second network has 5 ms delay before the bottleneck and 75 ms delay after it, while the third network is the opposite of the second. Statistical data for these VoIP calls can be found in tables 4.1, 4.11 and 4.12. The results of these three simulations have no significant difference. It is obvious that the distance to the bottleneck has little influence on the performance of the VoIP calls. This is because the end-to-end delay in the network remains the same and neither the cross traffic or the VoIP call is influenced by the distance to the bottleneck.

Connections per second (HTTP)	Average packet loss per call (percent)	Standard deviation packet loss per call	Median packet loss per call	Minimum packet loss per call	Maximum packet loss per call	Minimum jitter (ms)	Maximum jitter (ms)	95th percentile minimum jitter (ms)	95th percentile maximum jitter (ms)
100	0.02	0.06	0.00	0.00	0.25	-12.460	20.036	-7.731	7.282
150	1.62	2.61	0.57	0.00	11.46	-12.460	22.582	-9.341	9.476
200	9.42	6.40	7.54	0.00	27.08	-12.428	26.761	-8.774	9.764
250	26.07	7.20	27.20	5.49	40.39	-12.364	19.348	-6.899	8.404
300	37.63	3.82	38.57	27.89	47.29	-11.667	18.841	-5.961	7.869
400	49.92	2.62	49.80	44.49	56.44	-11.616	19.038	-5.601	7.414
500	56.38	1.90	56.13	52.30	61.43	-10.813	23.490	-5.664	7.177
600	60.70	1.69	60.47	55.88	64.24	-11.945	19.711	-5.394	6.906
700	64.35	1.48	64.72	61.79	67.19	-11.703	19.255	-5.470	6.676
800	66.37	1.34	66.45	62.31	69.39	-11.473	17.845	-5.334	6.595
900	68.98	1.35	68.93	66.33	72.85	-11.454	20.246	-5.353	6.495

Table 4.12: Statistical data for a 32 kbps VoIP call with bottleneck placed 75 ms from the sender and 5 ms from the receiver in a network with 10 Mbps bandwidth, queueing method tail-drop and 200 packets queue length.

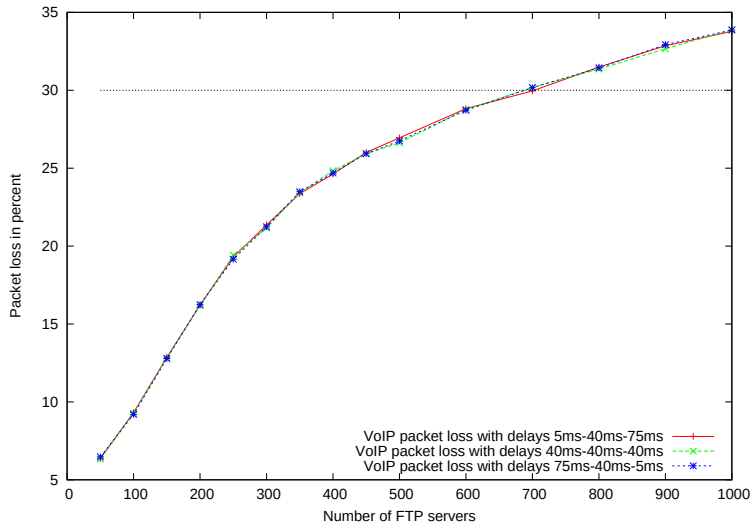

Figure 4.9: Packet loss plot for a 32 kbps VoIP call with cross traffic generated from FTP servers in a 10 Mbps varying distances to the bottleneck.

4.4.2 Varying the Distance to the Bottleneck with FTP cross traffic

Figure 4.9 shows a comparison between VoIP 32 kbps VoIP calls with varying distance to the bottleneck. The only difference from 4.4.1 is that the HTTP cross traffic is sub-

Number of FTP servers	Average packet loss per call (percent)	Standard deviation packet loss per call	Median packet loss per call	Minimum packet loss per call	Maximum packet loss per call	Minimum jitter (ms)	Maximum jitter (ms)	95th percentile minimum jitter (ms)	95th percentile maximum jitter (ms)
50	6.38	0.68	6.32	5.18	8.46	-12.460	24.740	-6.460	4.340
100	9.32	0.62	9.32	8.21	10.98	-12.460	18.740	-6.460	5.540
150	12.86	0.68	12.86	11.45	14.24	-12.460	12.740	-5.260	5.540
200	16.21	0.73	16.48	14.66	17.25	-11.260	15.140	-5.260	5.540
250	19.30	0.60	19.36	17.98	20.42	-11.260	14.004	-5.260	5.540
300	21.36	1.19	21.80	18.85	23.07	-11.260	18.836	-5.260	4.340
350	23.36	0.68	23.53	21.63	24.87	-11.260	16.340	-4.060	4.340
400	24.63	0.76	24.72	22.95	26.97	-11.260	14.004	-4.060	4.340
450	26.01	0.71	25.99	24.59	27.81	-10.060	13.972	-4.060	3.140
500	26.95	0.57	26.87	25.71	28.23	-10.060	12.804	-4.060	3.140
600	28.82	0.82	28.87	27.07	30.91	-11.260	10.372	-3.996	3.140
700	29.96	0.67	29.95	28.61	31.40	-8.860	9.172	-2.860	3.140
800	31.49	0.88	31.37	29.95	33.48	-7.660	9.140	-2.860	2.520
900	32.86	0.82	32.93	31.15	34.66	-7.660	10.340	-2.860	2.520
1000	33.78	0.87	33.84	32.05	35.81	-8.860	10.372	-2.860	1.940

Table 4.13: Statistical data for a 32 kbps VoIP call with bottleneck bottleneck placed 5 ms from the sender and 75 ms from the receiver in a network with 10 Mbps bandwidth, queueing method tail-drop and 200 packets queue length.

stituted with FTP cross traffic. This graph clearly shows that there is no difference between the packet loss for the VoIP calls with varying distance to the bottleneck and we can conclude that the choice of cross traffic has no influence on the results from varying the distance to the bottleneck. The statistical data for this plot can be found in tables 4.3, 4.13 and 4.14.

4.4.3 Varying the Distance to the Bottleneck Using VoIP with Silence Suppression

Fig. 4.10 shows a comparison between a bottleneck placed at equal distance from the receivers and senders, a bottleneck placed close to the senders and a bottleneck placed close to the receivers of the HTTP cross traffic and using silence suppression on the VoIP calls. If we compare this figure to fig. 4.8 we see a strong correlation, as expected.

Number of FTP servers	Average packet loss per call (percent)	Standard deviation packet loss per call	Median packet loss per call	Minimum packet loss per call	Maximum packet loss per call	Mini-mum jitter (ms)	Maxi-mum jitter (ms)	95th percentile minimum jitter (ms)	95th percentile maximum jitter (ms)
50	6.47	0.77	6.39	5.01	8.80	-12.460	22.340	-6.460	4.340
100	9.20	0.56	9.17	7.88	10.41	-12.460	16.340	-6.460	5.540
150	12.78	0.71	12.89	10.82	14.14	-12.460	16.340	-5.260	5.540
200	16.24	0.71	16.23	15.06	18.29	-11.260	12.740	-5.260	5.540
250	19.16	0.54	19.16	18.07	20.46	-11.260	13.972	-5.260	5.540
300	21.24	1.32	21.50	18.35	24.00	-11.260	13.972	-5.260	4.340
350	23.49	0.80	23.42	21.74	25.16	-11.260	15.172	-4.060	4.340
400	24.69	0.72	24.57	23.38	26.16	-12.428	16.436	-4.060	4.340
450	25.93	0.56	25.85	24.95	27.42	-11.260	11.540	-4.060	3.140
500	26.75	0.59	26.72	25.70	28.18	-11.260	12.740	-4.060	3.140
600	28.72	0.57	28.75	27.08	29.83	-11.260	10.340	-4.028	3.140
700	30.17	0.84	30.04	28.52	32.09	-8.860	11.540	-2.860	3.140
800	31.44	0.67	31.51	29.58	32.68	-8.860	11.540	-2.860	3.120
900	32.93	0.86	32.86	30.93	34.96	-8.860	10.340	-2.860	1.972
1000	33.87	0.81	33.96	32.01	35.55	-7.660	7.972	-2.860	1.940

Table 4.14: Statistical data for a 32 kbps VoIP call with bottleneck placed 75 ms from the sender and 5 ms from the receiver in a network with 10 Mbps bandwidth, queueing method tail-drop and 200 packets queue length.

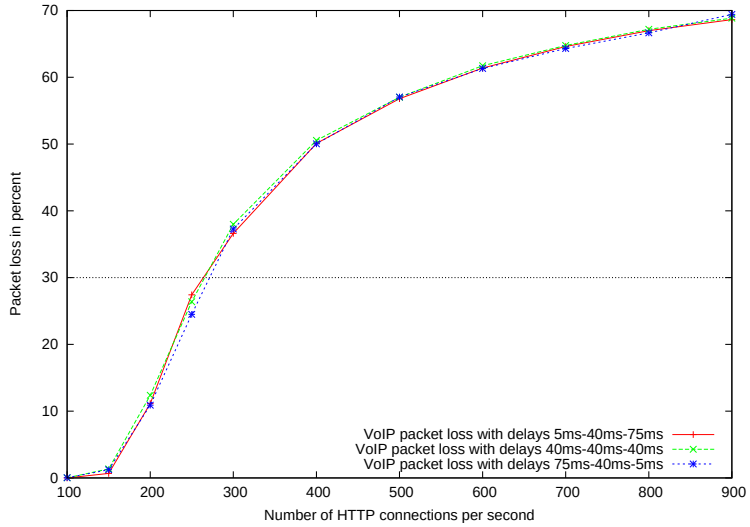

Figure 4.10: Packet loss with HTTP cross traffic with varying distance to the bottleneck using silence suppression.

Connections per second (HTTP)	Average packet loss per call (percent)	Standard deviation packet loss per call	Median packet loss per call	Minimum packet loss per call	Maximum packet loss per call	Minimum jitter (ms)	Maximum jitter (ms)	95th percentile minimum jitter (ms)	95th percentile maximum jitter (ms)
100	0.00	0.00	0.00	0.00	0.00	-12.460	22.924	-7.417	6.889
150	0.68	1.22	0.05	0.00	5.78	-12.460	26.304	-9.288	9.402
200	11.13	6.66	9.04	0.00	24.74	-12.460	23.621	-8.589	9.490
250	27.42	7.56	28.35	2.00	40.54	-12.364	21.154	-6.866	8.513
300	36.62	5.33	36.94	23.37	47.02	-12.007	19.523	-6.110	7.924
400	50.09	2.45	50.23	42.72	55.53	-10.907	19.283	-5.661	7.304
500	56.80	1.88	56.96	53.50	60.75	-10.652	17.190	-5.581	7.140
600	61.39	1.93	61.77	57.83	65.94	-11.918	21.023	-5.538	6.905
700	64.61	2.02	64.60	60.94	69.52	-10.876	18.804	-5.407	6.589
800	66.98	1.62	66.98	62.34	71.67	-10.720	16.600	-5.443	6.793
900	68.63	1.57	68.44	65.00	71.35	-11.034	17.661	-5.251	6.588

Table 4.15: Statistical data for a 32 kbps VoIP call using silence suppression and bottleneck placed 5 ms from the sender and 75 ms from the receiver in a network with 10 Mbps bandwidth, queueing method tail-drop and 200 packets queue length.

Connections per second (HTTP)	Average packet loss per call (percent)	Standard deviation packet loss per call	Median packet loss per call	Minimum packet loss per call	Maximum packet loss per call	Minimum jitter (ms)	Maximum jitter (ms)	95th percentile minimum jitter (ms)	95th percentile maximum jitter (ms)
100	0.03	0.14	0.00	0.00	0.95	-12.460	19.256	-7.824	7.338
150	1.23	1.75	0.32	0.00	7.30	-12.460	22.919	-9.452	9.559
200	10.89	6.29	10.09	0.84	23.63	-12.460	27.241	-8.676	9.691
250	24.50	6.19	24.91	13.67	35.83	-12.300	26.341	-7.100	8.568
300	37.28	5.34	37.86	21.45	47.41	-12.014	19.998	-6.140	7.862
400	50.03	3.06	49.74	42.12	55.94	-10.946	17.645	-5.655	7.166
500	57.04	2.04	57.18	51.74	61.03	-10.200	17.454	-5.635	7.154
600	61.32	1.86	61.17	57.26	65.55	-10.516	19.264	-5.563	6.642
700	64.31	2.24	64.20	59.22	70.99	-10.477	16.939	-5.500	6.838
800	66.65	1.23	66.70	63.97	69.35	-11.495	18.038	-5.377	6.667
900	69.42	1.49	69.52	65.09	72.09	-11.264	19.572	-5.331	6.458

Table 4.16: Statistical data for a 32 kbps VoIP call using silence suppression and bottleneck placed 75 ms from the sender and 5 ms from the receiver in a network with 10 Mbps bandwidth, queueing method tail-drop and 200 packets queue length.

It seems that the distance to the bottleneck still remains insignificant. The data for this plot can be found in tables 4.7, 4.15 and 4.16.

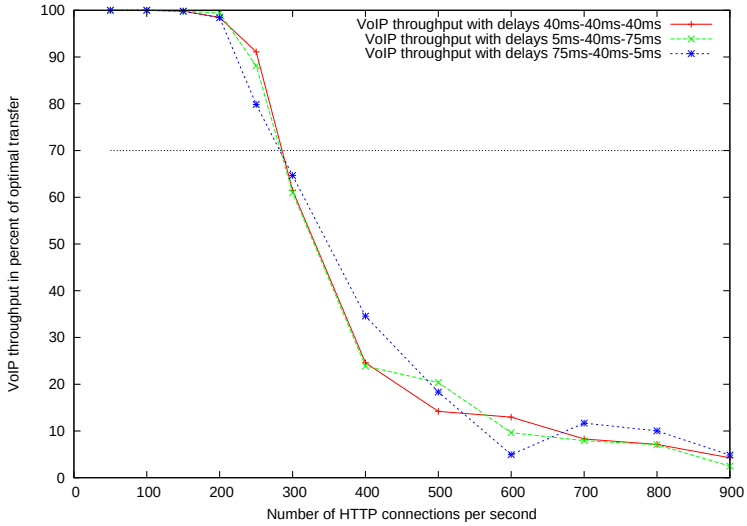

Figure 4.11: Comparison between 32 kbps TCP-based VoIP calls with varying distances to the bottleneck in a 10 Mbps network.

4.4.4 Varying the Placement of the Bottleneck with TCP-based VoIP

In fig. 4.11 we see the throughput for three different TCP-based VoIP calls. Each call is placed alone in different simulations subjected to the same cross traffic. The results indicate a marked difference between the calls, but no clear pattern as the bandwidth increases. This is probably because the loss of SYN packets makes such a significant impact on the performance of the TCP-based VoIP call. Because one lost packet makes such a big difference this simulation would have to be run several more times to get a good statistical basis. The statistical data for this plot can be found in tables 4.17, 4.9 and 4.18.

4.4.5 Queue Length

In this simulation, we tried varying the queue length defined in the routers. The queue length in our default simulation is 200. We have compared the packet drop in our default network with two other simulations where the queue length in one is 1000

Connections per second (HTTP) (HTTP)	Average throughput per call in percent of optimum	Standard deviation through-put put	Median throughput per call	Minimum throughput per call	Maximum throughput per call
50	100.00	0.00	100.00	100.00	100.00
100	99.99	0.04	100.00	99.76	100.00
150	99.78	0.29	99.97	99.22	100.00
200	99.43	0.21	99.34	99.11	100.00
250	88.15	24.33	99.31	0.00	99.52
300	60.97	34.50	68.91	0.00	99.52
400	23.88	23.13	15.68	0.00	71.42
500	20.37	19.86	9.58	0.00	60.50
600	9.64	15.40	0.00	0.00	50.36
700	7.92	12.21	0.87	0.00	47.16
800	7.05	10.99	1.74	0.00	38.47
900	2.49	4.64	0.00	0.00	17.72

Table 4.17: Statistical data for 32 kbps TCP-based VoIP call with bottleneck placed 5 ms from the sender and 75 ms from the receiver in a network with 10 Mbps bandwidth, queueing method tail-drop and 200 packets queue length.

Connections per second (HTTP) (HTTP)	Average throughput per call in percent of optimum	Standard deviation through-put put	Median throughput per call	Minimum throughput per call	Maximum throughput per call
50	100.00	0.00	100.00	100.00	100.00
100	100.00	0.03	100.00	99.82	100.00
150	99.82	0.25	100.00	99.22	100.00
200	98.40	5.56	99.37	61.20	99.64
250	79.86	29.93	99.01	0.00	99.52
300	64.64	28.17	70.96	1.75	99.52
400	34.57	27.55	37.26	0.00	89.51
500	18.34	18.72	12.24	0.00	61.92
600	4.95	9.63	0.00	0.00	35.39
700	11.69	14.62	2.62	0.00	50.90
800	10.02	12.36	3.50	0.00	39.52
900	4.86	9.62	0.00	0.00	39.58

Table 4.18: Statistical data for 32 kbps TCP-based VoIP call with bottleneck placed 75 ms from the sender and 5 ms from the receiver in a network with 10 Mbps bandwidth, queueing method tail-drop and 200 packets queue length.

Connections per second (HTTP)	Average packet loss per call (percent)	Standard deviation packet loss per call	Median packet loss per call	Minimum packet loss per call	Maximum packet loss per call	Minimum jitter (ms)	Maximum jitter (ms)	95th percentile minimum jitter (ms)	95th percentile maximum jitter (ms)
100	0.00	0.00	0.00	0.00	0.00	-12.460	18.763	-7.544	7.008
150	0.05	0.19	0.00	0.00	1.23	-12.460	26.701	-9.571	10.044
200	3.49	4.43	2.13	0.00	18.26	-12.428	25.190	-9.577	10.713
250	14.32	7.16	13.28	2.45	26.86	-12.428	29.050	-8.566	10.135
300	26.21	5.13	26.35	14.33	35.40	-12.268	27.969	-7.316	9.071
400	42.64	3.23	42.36	33.82	48.51	-11.980	22.688	-6.093	8.015
500	50.72	2.25	50.51	46.38	57.08	-10.824	20.551	-5.875	7.772
600	56.32	1.97	56.24	52.19	60.08	-10.793	21.091	-5.674	7.513
700	59.80	1.64	60.11	53.23	62.71	-11.815	26.765	-5.606	7.355
800	63.08	1.37	63.13	60.44	65.57	-10.507	20.629	-5.564	7.105
900	65.34	1.37	65.35	62.85	69.58	-11.087	21.278	-5.529	6.992

Table 4.19: Statistical data for a 32 kbps VoIP call with bottleneck placed equidistantly from sender and receiver in a network with 10 Mbps bandwidth, queueing method tail-drop and 1000 packets queue length.

Connections per second (HTTP)	Average packet loss per call (percent)	Standard deviation packet loss per call	Median packet loss per call	Minimum packet loss per call	Maximum packet loss per call	Minimum jitter (ms)	Maximum jitter (ms)	95th percentile minimum jitter (ms)	95th percentile maximum jitter (ms)
100	0.00	0.00	0.00	0.00	0.00	-12.460	23.144	-7.476	6.868
150	0.00	0.00	0.00	0.00	0.00	-12.460	23.846	-9.555	9.852
200	0.51	1.56	0.00	0.00	7.30	-12.460	30.542	-9.890	11.048
250	6.75	5.01	6.10	0.00	17.22	-12.460	30.472	-9.380	11.401
300	16.8	4.41	17.19	6.23	27.57	-12.396	30.863	-8.510	10.564
400	34.89	4.09	34.88	26.43	44.30	-12.396	25.303	-6.982	8.950
500	45.14	2.39	44.90	39.02	51.38	-11.372	21.141	-6.254	8.425
600	51.56	2.16	51.01	46.31	55.37	-11.671	19.476	-6.044	8.070
700	56.41	1.89	56.80	52.01	61.14	-11.883	20.317	-5.882	7.776
800	60.30	1.34	60.41	57.47	63.79	-11.191	20.068	-5.760	7.557
900	62.98	1.43	62.83	58.76	65.57	-11.606	19.489	-5.755	7.464

Table 4.20: Statistical data for a 32 kbps VoIP call with bottleneck placed equidistantly from sender and receiver in a network with 10 Mbps bandwidth, queueing method tail-drop and 2000 packets queue length.

Connections per second(HTTP)	Average retransmissions per packet	Standard Deviation	Median	Minimum	Maximum
100	0.00	0.00	0.00	0.00	0.00
150	0.00	0.00	0.00	0.00	0.01
200	0.04	0.05	0.02	0.00	0.21
250	0.17	0.09	0.15	0.03	0.34
300	0.35	0.09	0.35	0.16	0.50
400	0.70	0.08	0.70	0.49	0.90
500	0.96	0.07	0.94	0.83	1.16
600	1.22	0.09	1.23	1.05	1.41
700	1.42	0.07	1.42	1.15	1.55
800	1.65	0.07	1.65	1.44	1.79
900	1.84	0.08	1.84	1.62	2.01

Table 4.21: Statistical data for TCP retransmissions in a network with 10 Mbps bandwidth, queueing method tail-drop and 1000 packets queue length.

Connections per second(HTTP)	Average retransmissions per packet	Standard Deviation	Median	Minimum	Maximum
100	0.00	0.00	0.00	0.00	0.00
150	0.00	0.00	0.00	0.00	0.00
200	0.01	0.02	0.00	0.00	0.09
250	0.07	0.06	0.07	0.00	0.20
300	0.20	0.06	0.21	0.07	0.38
400	0.51	0.08	0.51	0.36	0.72
500	0.78	0.07	0.77	0.60	0.94
600	0.99	0.07	0.98	0.82	1.15
700	1.22	0.07	1.23	1.07	1.35
800	1.44	0.06	1.43	1.33	1.60
900	1.64	0.06	1.63	1.51	1.76

Table 4.22: Statistical data for TCP retransmissions in a network with 10 Mbps bandwidth, queueing method tail-drop and 2000 packets queue length.

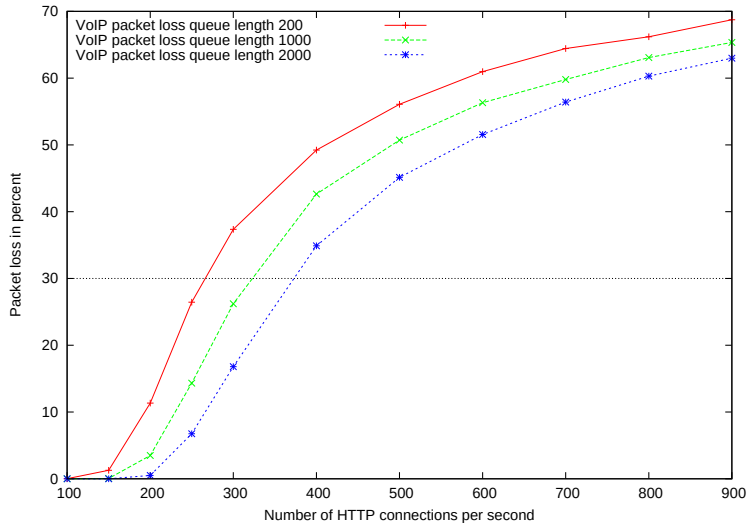

Figure 4.12: Comparison of packet loss for a 32 kbps VoIP call in networks with queue length 200, 1000 and 2000.

packets and the queue length is 2000 packets in the other. The resulting plot in figure 4.12 shows a comparison between the simulations with increased queue length and the one from our default simulation (figure 4.1). This graph shows a significant difference in favour of the networks with increased queue length. The statistical data for these simulations can be found in 4.1, 4.19 and 4.20. There is no significant difference in jitter for the different simulations, but the end-to-end delay for the packets will increase proportionately with the queue length when there is a lot of traffic in the network. If we compare the retransmission data table from our default experiment (table 4.2) with the corresponding tables for the network with queue lengths of a 1000 packets (table 4.21) and 2000 packets (table 4.22), we see that there is less contention between the TCP flows. All data on retransmitted TCP packets are lower for the increased queue lengths.

4.4.6 Queueing Method

The queueing method used in the simulation is as expected the most significant change that can be made in the network with regards to favoring UDP packets. The basic queueing method used in my simulations is tail-drop. While this can be considered a fair queueing method, it tries to deal with congestion when it arises as opposed to

Connections per second (HTTP)	Average packet loss per call (percent)	Standard deviation packet loss per call	Median packet loss per call	Minimum packet loss per call	Maximum packet loss per call	Minimum jitter (ms)	Maximum jitter (ms)	95th percentile minimum jitter (ms)	95th percentile maximum jitter (ms)
100	0.01	0.03	0.00	0.00	0.19	-12.460	20.032	-7.276	6.865
150	0.09	0.08	0.06	0.00	0.31	-12.460	19.761	-7.587	7.664
200	0.22	0.11	0.19	0.00	0.44	-12.332	18.879	-6.437	7.060
250	0.25	0.10	0.25	0.06	0.44	-12.364	16.865	-5.943	6.672
300	0.15	0.10	0.13	0.00	0.44	-12.364	17.088	-7.104	6.409
400	0.14	0.08	0.13	0.00	0.38	-12.364	18.016	-10.459	7.804
500	0.10	0.08	0.06	0.00	0.31	-12.332	19.269	-10.574	9.866
600	0.12	0.09	0.13	0.00	0.38	-12.236	23.764	-10.465	11.483
700	0.11	0.08	0.13	0.00	0.31	-12.204	21.957	-10.349	12.773
800	0.10	0.08	0.06	0.00	0.31	-12.111	24.296	-10.169	13.901
900	0.12	0.08	0.13	0.00	0.31	-12.108	23.808	-9.999	14.539

Table 4.23: Statistical data for a 32 kbps VoIP call with bottleneck placed equidistantly from sender and receiver in a network with 10 Mbps bandwidth, queueing method RED and 200 packets queue length.

Connections per second(HTTP)	Average retransmissions per packet	Standard Deviation	Median	Minimum	Maximum
100	0.01	0.01	0.00	0.00	0.03
150	0.04	0.03	0.03	0.00	0.18
200	0.14	0.06	0.14	0.03	0.26
250	0.29	0.06	0.30	0.13	0.39
300	0.40	0.05	0.39	0.25	0.49
400	0.55	0.03	0.55	0.50	0.62
500	0.63	0.02	0.63	0.58	0.68
600	0.68	0.02	0.69	0.62	0.72
700	0.73	0.02	0.73	0.69	0.75
800	0.76	0.02	0.76	0.72	0.80
900	0.78	0.02	0.78	0.75	0.81

Table 4.24: Statistical data for TCP retransmissions in a network with 10 Mbps bandwidth, queueing method RED and 200 packets queue length.

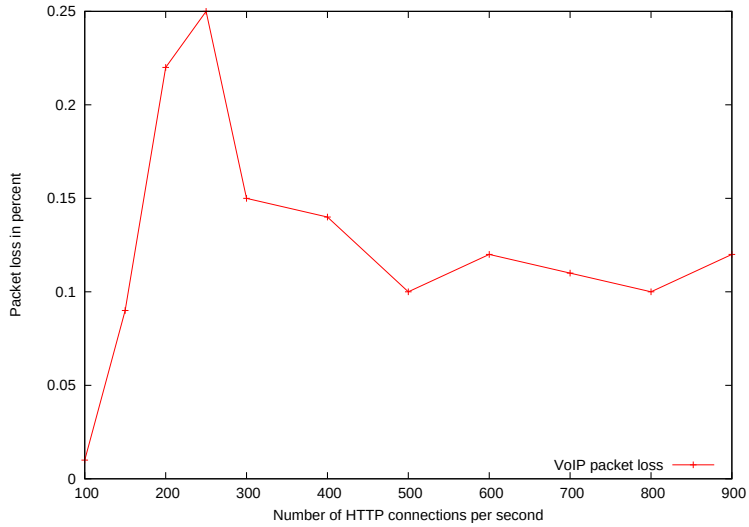

Figure 4.13: Packet loss for a 32 kbps VoIP call in a network with queueing method RED.

Random Early Drop (RED) which tries to avoid it by dropping random packets as the bandwidth consumption grows closer to the point of congestion. Figure 4.13 shows the dropped packets in a network using the queueing method RED and the corresponding statistical data can be found in table 4.23. If we compare this graph and table with the ones in figure 4.1 and table 4.1, we see that the queuing method clearly favors the UDP packets. Pay attention to the scales on the axis. Although we had expected increased throughput for the VoIP call in the network using RED, the improvement was a lot higher than expected. Although there are drawback. If we look at the jitter data for these simulations, we see an increase in jitter for the VoIP call in the network using RED. This is probably due to the fact that when RED drops packets, there is still available bandwidth in the network which the TCP packets will contend for. This causes the traffic in the network to fluctuate. If we compare the data tables for TCP retransmissions in our default simulation (table 4.2) with the corresponding table for the network using RED (table 4.24), we see a significant drop in retransmissions. This is a result from the fact that RED keeps the TCP flows down and since packets are lost randomly instead of in bursts, fewer retransmissions are required.

Connections per second (HTTP)	Average packet loss per call (percent)	Standard deviation packet loss per call	Median packet loss per call	Minimum packet loss per call	Maximum packet loss per call	Minimum jitter (ms)	Maximum jitter (ms)	95th percentile minimum jitter (ms)	95th percentile maximum jitter (ms)
1000	0.00	0.00	0.00	0.00	0.00	-7.077	6.417	-1.806	1.761
1500	0.03	0.08	0.00	0.00	0.31	-9.066	8.946	-3.655	3.567
2000	13.71	3.72	13.75	7.25	20.73	-7.935	8.280	-3.172	3.681
2500	32.56	1.83	32.36	28.90	36.36	-4.828	6.225	-2.328	2.753
3000	42.07	1.82	41.91	37.34	45.01	-4.570	6.354	-2.137	2.462
4000	52.11	1.36	52.28	49.40	54.60	-4.121	5.092	-2.074	2.287
5000	57.99	0.95	58.12	56.12	59.32	-4.104	5.012	-2.027	2.224
6000	62.12	1.26	62.07	60.01	64.47	-4.197	5.772	-1.976	2.215
7000	64.81	1.47	64.32	62.86	67.92	-4.228	5.184	-1.931	2.208

Table 4.25: Statistical data for a 32 kbps VoIP call with bottleneck placed equidistantly from sender and receiver in a network with 100 Mbps bandwidth, queueing method tail-drop and 2000 packets queue length.

Connections per second(HTTP)	Average retransmissions per packet	Standard Deviation	Median	Minimum	Maximum
1000	0.00	0.00	0.00	0.00	0.00
1500	0.00	0.00	0.00	0.00	0.00
2000	0.10	0.03	0.09	0.04	0.16
2500	0.35	0.03	0.34	0.31	0.41
3000	0.57	0.03	0.56	0.50	0.63
4000	0.91	0.03	0.92	0.85	0.96
5000	1.18	0.02	1.18	1.14	1.21
6000	1.43	0.03	1.44	1.35	1.49
7000	1.67	0.02	1.67	1.63	1.72

Table 4.26: Statistical data for TCP retransmissions in a network with 100 Mbps bandwidth, queueing method tail-drop and 2000 packets queue length.

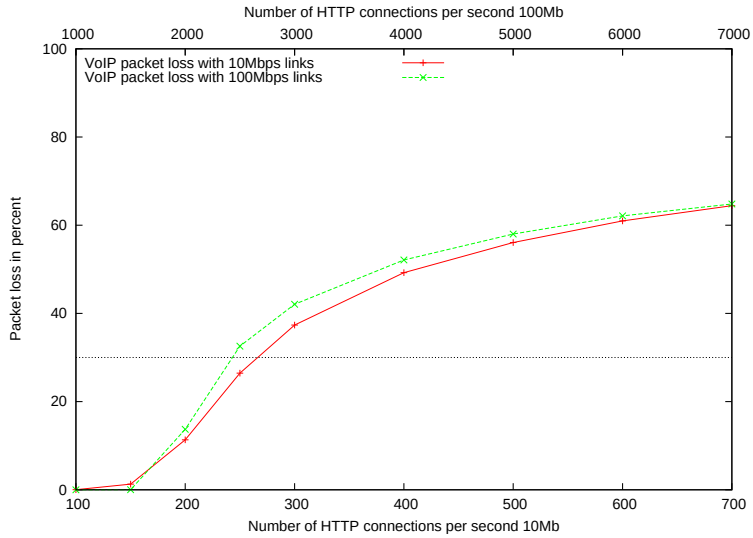

Figure 4.14: Packet loss for a 32 kbps VoIP call in a 10 Mbps network compared to packet loss in a 100 Mbps network.

4.4.7 Bandwidth

This simulation is to demonstrate the scalability of the simulations in general. We have increased the bandwidth to 100 Mbps, number of connections per second with a factor of ten, number of PackMime-HTTP nodes to 100 and the queue length to 2000. The VoIP bandwidth demand remains the same at 32 kbps. Figure 4.14 shows a comparison between the packet loss in the default network and the packet loss in the 100 Mbps network just described. The cross traffic for the 10 Mbps network is indicated by the lower x-axis and the cross traffic for the 100 Mbps network is indicated by the upper x-axis. This simulation is, as mentioned in 4.1, the result of only 20 runs at each level of cross traffic. As we can see the packet loss is almost identical in the up scaled simulation. It is in fact slightly higher in the 100 Mbps network. This is more due to the fact that the up-scaling of the HTTP traffic results in even more contention between the TCP flows and consequently utilize a higher percentage of the bandwidth in the network. The statistical data for these simulations can be found in table 4.1 and table 4.25. If we look at jitter here, we see a significant improvement in the 100 Mbps network. This is most likely for the same reason as the increased packet loss. The network is constantly utilized at a higher percentage and the thousand of TCP connections which make the delays for the VoIP calls more constant. When we take a look at the retrans-

Connections per second (HTTP)	Average packet loss per call (percent)	Standard deviation packet loss per call	Median packet loss per call	Minimum packet loss per call	Maximum packet loss per call	Minimum jitter (ms)	Maximum jitter (ms)	95th percentile minimum jitter (ms)	95th percentile maximum jitter (ms)
1000	0.00	0.00	0.00	0.00	0.00	-5.252	4.564	-1.549	1.517
1500	0.06	0.25	0.00	0.00	1.11	-5.188	5.456	-2.511	2.427
2000	9.85	3.50	10.10	2.40	14.84	-5.086	5.977	-2.290	2.514
2500	27.86	3.14	28.30	21.06	34.30	-3.441	4.671	-1.621	1.923
3000	37.60	2.10	37.25	34.74	40.98	-3.039	4.492	-1.502	1.769
4000	48.57	1.16	48.37	46.78	51.62	-3.068	4.058	-1.428	1.632
5000	55.24	1.22	55.59	52.98	57.88	-3.245	4.788	-1.398	1.600
6000	59.47	1.12	59.35	57.54	61.14	-2.891	3.815	-1.408	1.608
7000	62.70	1.35	62.47	60.47	66.15	-3.382	4.030	-1.391	1.565

Table 4.27: Statistical data for a 64 kbps VoIP call with bottleneck placed equidistantly from sender and receiver in a network with 100 Mbps bandwidth, queueing method tail-drop and 2000 packets queue length.

mission data table for this simulation (table 4.25) and compare it to the one from our default simulation (table 4.2), there is almost no difference. We think this indicates that our simulation is valid for higher bandwidths, except for the data collected on jitter.

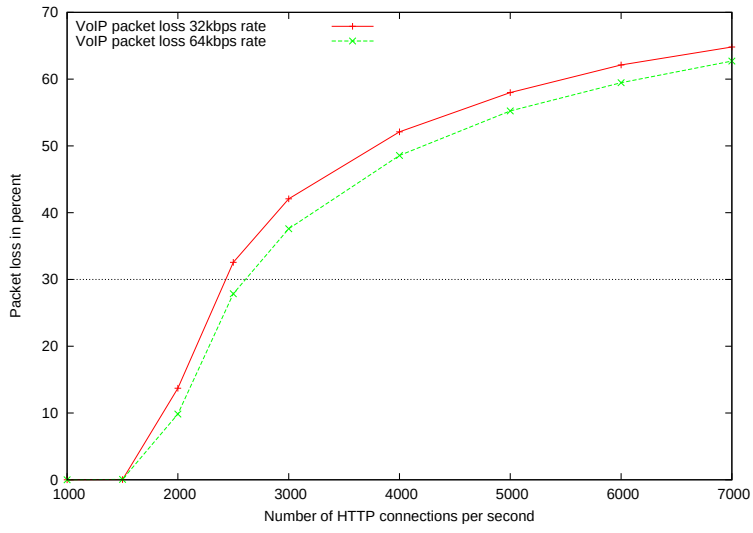

Figure 4.15: Comparison between 32 kbps and 64 kbps VoIP calls in a 100 Mbps network.

4.4.8 Varying the Bandwidth Used by a VoIP Call in a 100 Mbps Network

In this section we present a comparison between the packet loss of a 32 kbps VoIP call and the packet loss of a 64 kbps VoIP call. Both calls have been placed in the higher bandwith-scenario 100 Mbps network presented in section 4.4.7. The results are seen in fig. 4.15 and if we compare this figure with 4.3 we see that the higher bandwidth does not make any significant change on the packet loss for the VoIP bandwidths except for the slight increase in packet loss described in 4.4.7. The 64 kbps VoIP call still performs better than the 32 kbps VoIP call. The data for this graph can be found in tables 4.25 and 4.27. Statistical data for this graph can be found in 4.25 and 4.27.

4.4.9 Repercussions of Network Changes

Changing variables in the network without looking at the repercussions on TCP traffic would be shortsighted. We have therefore measured and compared the efficiency of the bandwidth sharing of TCP traffic in the bottleneck in the networks using different queueing methods, the networks with different bandwidth and the networks using different queue lengths.

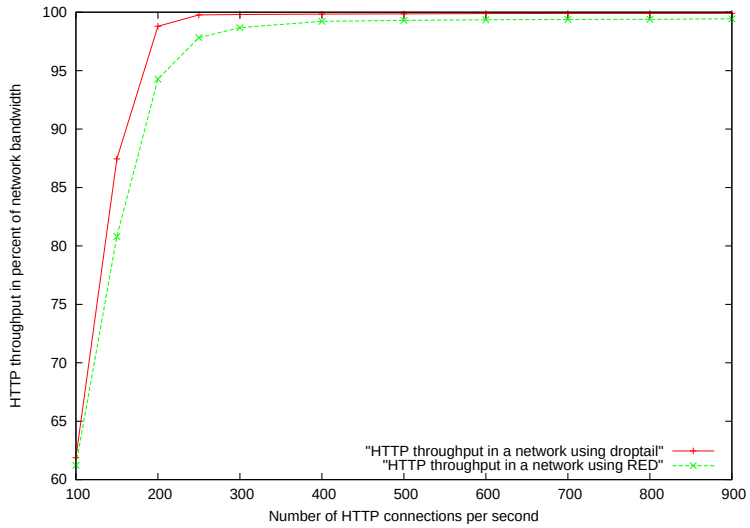

Figure 4.16: Comparison between HTTP cumulative in a network with queuing method RED and a network with queueing method tail-drop.

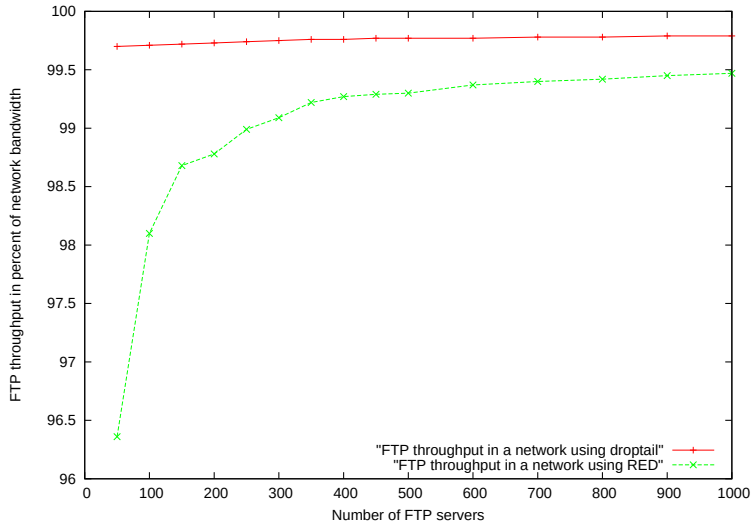

Figure 4.17: Comparison between FTP cumulative in a network with queueing method RED and a network with queueing method tail-drop.

We have done two cumulative comparisons between the network using queueing method RED and the network using queueing method tail-drop. The first is a comparison between the cumulative for HTTP, which can be seen in figure 4.16, and the second one is a comparison between the cumulative for FTP, which can be seen in figure 4.17. From both plots we see that the reduced packet loss for the VoIP call in the network with queueing method RED comes at a small prize. The cumulative for both traffic types is lower in the network with queueing method RED, but not by much.

In figure 4.18, we see a comparison of HTTP cumulative in a network with a queue length of 200 packets, one with a queue length of 1000 packets and one with a queue length of 2000 packets. This plot clearly show that there is no difference between the cumulative for the three networks.

Finally we did a comparison between the HTTP cumulative of the 10 Mbps network and the 100 Mbps network. The HTTP cumulative data for these networks can be seen in table 4.28 for the 10 Mbps network and in table 4.29 for the 100 Mbps network. As we can see the 100 Mbps network shows a slightly higher cumulative in percent of network bandwidth, which accounts for the results discussed in 4.4.7.

Connections per second (HTTP)	Average cumulative in percent (HTTP)	Standard deviation cumulative	Median cumulative	Minimum cumulative	Maximum cumulative
100	61.88	9.99	62.47	44.87	96.36
150	87.45	9.99	87.20	63.82	99.72
200	98.80	2.33	99.71	90.01	99.78
250	99.77	0.02	99.77	99.70	99.81
300	99.80	0.01	99.80	99.77	99.83
400	99.84	0.01	99.84	99.82	99.86
500	99.86	0.01	99.86	99.84	99.87
600	99.88	0.01	99.87	99.86	99.89
700	99.89	0.01	99.89	99.87	99.90
800	99.89	0.01	99.89	99.88	99.90
900	99.90	0.01	99.90	99.89	99.91

Table 4.28: Statistical data for HTTP cumulative in a network with 10 Mbps bandwidth, queueing method tail-drop and 200 packets queue length.

Connections per second (HTTP)	Average cumulative in percent (HTTP)	Standard deviation cumulative	Median cumulative	Minimum cumulative	Maximum cumulative
1000	60.74	5.91	60.48	48.62	70.44
1500	90.85	4.88	91.12	79.44	99.36
2000	99.97	0.00	99.97	99.97	99.97
2500	99.98	0.00	99.98	99.98	99.98
3000	99.98	0.00	99.98	99.98	99.98
4000	99.98	0.00	99.98	99.98	99.99
5000	99.99	0.00	99.99	99.99	99.99
6000	99.99	0.00	99.99	99.99	99.99
7000	99.99	0.00	99.99	99.99	99.99

Table 4.29: Statistical data for HTTP cumulative in a network with 100 Mbps bandwidth, queueing method tail-drop and 2000 packets queue length.

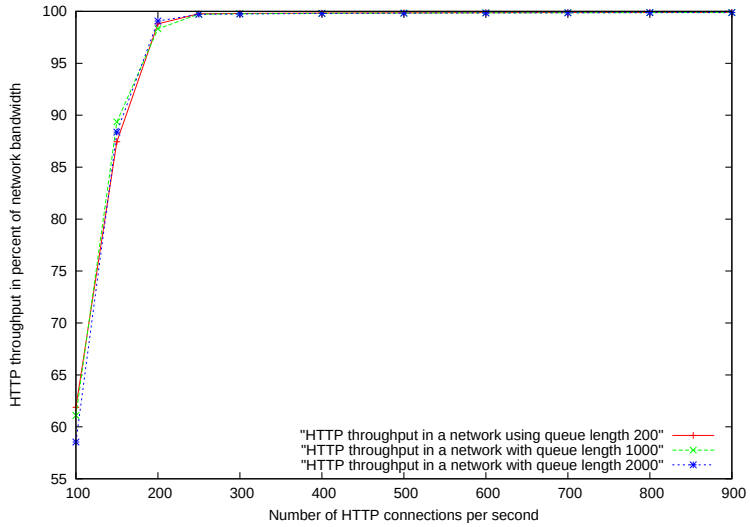

Figure 4.18: Comparison between HTTP cumulative in a network using a queue length of 200, 1000 and 2000.

4.5 Summary

In this chapter, we have presented the results of our simulations. We found that the threshold for significant packet loss was around 250 HTTP connections per second in a 10 Mbps network and around 2500 HTTP connections per second in a 100 Mbps network. The threshold in the 10 Mbps network with FTP cross traffic was around 800 FTP servers. The changes in the endpoints that changed TCP's influence on the VoIP call was the rate of the VoIP calls and the choice of Transport protocol. The use of silence suppression had a small negative effect. The changes in the network that made a positive difference to TCP's influence on the VoIP call were the queue length and the queuing method. Changing the bandwidth had a marginal effect, while the variations in the distance to the bottleneck had no effect at all.

Chapter 5

Conclusions and Future Work

In this chapter, we will give a short summary, take another look at our problem statement and see if we have found any answers. Finally, we will give some thoughts about future research.

5.1 Summary

In this thesis, we have investigated the influence of TCP flows on interactive multimedia traffic. This was done through a series of simulations in the network simulator NS-2. The topology for the simulation network was a set of network links meeting at a mutual bottleneck. The effects of TCP cross traffic on VoIP calls was thoroughly tested with, many alternations in the endpoints and the network itself. We found a threshold for extremely disruptive cross traffic based on iLBC at 30 percent packet loss. The amount of web traffic required to reach this threshold was significantly lower than for FTP traffic.seriously disrup a V We believe our findings are relevant for other kinds of interactive multimedia traffic as well. Multi-player online games which we describe in 2.2.2, have the same traffic patterns as VoIP, but stricter demands on latency and jitter. VoD described in 2.2.3 might be a little more robust depending on the buffer size and codecs used.

The changes made in the endpoints that affected TCP's influence on the VoIP traffic were the choice of transport protocol and the rate of the VoIP calls. Silence suppression

made a little effect for the worse, i.e., it reduced the VoIP bandwidth consumption, but the received quality was reduced due to higher loss.

The change of queuing methods from tail-drop to RED made the packet loss for the VoIP call drop to almost zero. The increase in queue length improved on the packet loss, but it also introduced higher latency. The distance to the bottleneck proved to have no effect at all. Our simulations in the 100Mbps version of our network shows that our results are valid for higher bandwidth as well.

5.2 Conclusion

If we take a look at our problem statement and questions from 1.2, we theorized many interactive multimedia applications that provide adequate QoS at the time might degrade in the future as more traffic is introduced in the Internet. Based on our finding, we think this is very likely.

5.2.1 What kind of traffic scenarios did disrupt a VoIP call?

We have tried to disrupt a VoIP call with both FTP traffic and web traffic. Both traffic types were able to disrupt a VoIP call. It is reasonable to assume that any kind of traffic in sufficient amounts would be able to do this, but are all traffic scenarios plausible? As we saw with the FTP cross traffic in section 4.3.1, the amount of FTP servers in a 10 Mbps network to get 30 percent packet loss, was 700. This traffic scenario is not plausible at all. The results in section 4.2, which needed only 250 HTTP connections per second, on the other hand is quite plausible. Based on our simulations with FTP and HTTP cross traffic, it is obvious that web traffic is the most threatening traffic of the two. This kind of traffic is unfortunately, and obviously, abundant in the Internet.

5.2.2 How badly congested was the network before serious disruption occurs?

Our results from chapter 4 indicate that the network has to be quite badly congested before serious disruption occurs for the VoIP application. UDP holds no special status in a router and is subject to the same kind of packet loss as all other packets. This means that for packet loss of 30 percent in a UDP stream to occur, all other streams loose just as much. It is still worth considering that badly congested does not mean unlikely congested. This means that unless the network technology is upgraded and replaced to support the growing traffic in the Internet alternative technology for interactive multimedia traffic must be researched and implemented.

5.2.3 What modifications to end-point and network variables improved the situation?

From our simulation, we can draw the following conclusions: TCP may be a very good alternative to UDP as a transport protocol if it is modified by late data choice [16] or retransmission latency reduction [14] [12]. The most significant changes are the queue length and queuing method. Increasing the queue length is quite effective in reducing the packet loss in the network, although it introduces some extra delay in the network. The main problems with increasing the queue length is the cost of computer memory and the increased latency when the queues fill up. Routers require extremely fast memory, and extremely fast is of course extremely expensive. The most effective variation in the network was the introduction of the queueing method RED. Although there was a slight decrease in network utilization. RED is an easy and cheap algorithm to implement, and it would make a big difference.

5.3 Future Work

There are still a lot of alternatives to the modifications done in this thesis, and many are worth exploring. Simulations using other queueing methods could provide interesting results. Other types of cross-traffic could be just as devastating. There is a lot of P2P

traffic in the Internet and BitTorrent shortly described in 2.3.3 shows many of the characteristics of web traffic. It is also one of the most popular P2P programs today [19]. Finally, it is worth mentioning that although NS-2 is a powerful tool, it is still just a network simulator, and it can not replace real life tests. Experiments with a real network and real nodes should be performed to verify our results.

Appendix A

Simulation Source Code

A.1 VoIP-simulation.tcl

```
1  # Simulation script
2
3  # useful constants
4  set CLIENT 0
5  set SERVER 1
6
7  #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
8  # Usage
9  #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
10
11 proc usage {} {
12     puts stderr {usage: ns VoIP-simulation.tcl [variable value]
13         defaults:
14         duration        30        - length of measured simulation (seconds)
15         warmup          10        - warmup interval (seconds)
16         rate            0         - number of new connections / second
17         packmime-count  0         - number of PackMime clouds
18         queue-method    DropTail  - DropTail, RED, ...
19         queue-length    200       - queue length (packets)
20         pre-delay       40ms      - propagation delay before bottleneck
21         link-delay      40ms      - propagation delay on network links
22         post-delay      40ms      - propagation delay after bottleneck
23         link-bw         10Mb      - bandwidth on network links
24         voip-size       50        - payload size VoIP calls
25         voip-rate       32kb      - Bandwith rate VoIP calls
26         tcp-voip        0         - number of TCP based VoIP calls
27         udp-voip        0         - number of UDP based VoIP calls
28         ftp-servers     0         - number of ftp server/client pairs
29         window          2000      - max TCP window size (packets)
30         segsize         1460      - default TCP segment size (bytes)
31         loss-rate       0         - maximum packet loss rate
32         run             1         - experiment run number
33         debug           0         - debug level
34         quiet           0         - don't output status messages?
35         datadir         data      - directory to store output
36         gzip            1         - gzip trace-queue
37     }
38     exit 1
39 }
```

```
40
41  #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
42  # Command-Line Option Default Values
43  #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
44
45  proc default_options {} {
46      global opts opt_wants_arg
47
48      set raw_opt_info {
49
50          duration 30
51          warmup 10
52
53          # rate -- total number of connections/s
54          # packmime-count -- number of PackMime instances
55          rate 0
56          packmime-count 0
57
58          # network topo
59          queue-method DropTail
60          queue-length 200
61          pre-delay 40ms
62          link-delay 40ms
63          post-delay 40ms
64          link-bw 10Mb
65          voip-size 50
66          voip-rate 32kb
67          tcp-voip 0
68          udp-voip 0
69          ftp-servers 0
70
71          # TCP
72          window 2000
73          segsize 1460
74
75          # experiment run number
76          run 1
77
78          # data collection
79          datadir data
80          quiet 0
81          debug 0
82          gzip 1
83      }
84
85      # parses the default options and creates the $opts array
86      while {$raw_opt_info != ""} {
87          if {![regexp "^\[^\n\]*\n" $raw_opt_info line]} {
88              break
89          }
90          regsub "^\[^\n\]*\n" $raw_opt_info {} raw_opt_info
91          set line [string trim $line]
92          if {[regexp "^\[ \t\]*#" $line]} {
93              continue
94          }
95          if {$line == ""} {
96              continue
97          } elseif [regexp {^([^ ]+)[ ]+([^ ]+)$} $line dummy key value] {
98              set opts($key) $value
99              set opt_wants_arg($key) 1
100         } else {
101             set opt_wants_arg($key) 0
102             # die "unknown stuff in raw_opt_info\n"
```

```
103         }
104     }
105 }
106
107 #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
108 # Process Command Line Arguments
109 #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
110
111 proc process_args {} {
112     global argc argv opts opt_wants_arg
113
114     default_options
115     for {set i 0} {$i < $argc} {incr i} {
116         set key [lindex $argv $i]
117         if {$key == "-?" || $key == "--help" || $key == "-help" || $key == "-h"} {
118             usage
119         }
120         regsub {^-} $key {} key
121         if {![info exists opt_wants_arg($key)]} {
122             puts stderr "unknown option $key";
123             usage
124         }
125         if {$opt_wants_arg($key)} {
126             incr i
127             set opts($key) [lindex $argv $i]
128         } else {
129             set opts($key) [expr !opts($key)]
130         }
131     }
132 }
133
134 #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
135 # Setup Simulator
136 #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
137
138 remove-all-packet-headers;              # removes all packet headers
139 add-packet-header IP TCP;               # adds TCP/IP headers
140 set ns [new Simulator];                 # instantiate the Simulator
141 $ns use-scheduler Heap;                 # use the Heap scheduler
142 set node_count 2;                       # used for keeping track of assigned nodes
143
144 #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
145 # Setup Topology
146 #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
147 #           udp-v  udp-v
148 #     ftp-s \ |       | / ftp-c
149 #    web-s -- 0 ----- 1 -- web-c
150 #           / |       | \
151 #           tcp-v    tcp-v
152
153 proc create_topology {link_bw link_delay pre_delay post_delay queue_method queue_length} {
154     global ns n num_node node_count
155
156     # Creating the required number
157     for {set i 0} {$i < $num_node} {incr i} {
158         set n($i) [$ns node]
159         puts "node"
160     }
161
162     # EDGES (from-node to-node length a b):
163     $ns duplex-link $n(0) $n(1) $link_bw $link_delay $queue_method
164     $ns queue-limit $n(0) $n(1) $queue_length
165     $ns queue-limit $n(1) $n(0) $queue_length
```

```
166     $ns duplex-link-op $n(0) $n(1) orient right
167
168     # Connect nodes and set queue-limit
169     for {set i 2} {$i < $num_node} {incr i} {
170         $ns duplex-link $n($i) $n(0) $link_bw $pre_delay $queue_method
171         $ns queue-limit $n($i) $n(0) $queue_length
172         $ns queue-limit $n(0) $n($i) $queue_length
173         $ns duplex-link-op $n($i) $n(0) orient right
174         incr i
175         $ns duplex-link $n($i) $n(1) $link_bw $post_delay $queue_method
176         $ns queue-limit $n($i) $n(1) $queue_length
177         $ns queue-limit $n(1) $n($i) $queue_length
178         $ns duplex-link-op $n($i) $n(1) orient left
179     }
180 }
181 # end of create_topology
182
183 proc setup-tcp-voip {tcp_voip voip_size voip_rate} {
184     #Setup a TCP-VoIP connection
185     global ns n node_count tcpvoip
186
187     for {set i 0} {$i < $tcp_voip} {incr i} {
188         puts "One tcp-voip"
189         #Setup a FullTCP connection
190         set tcp($i) [new Agent/TCP/FullTcp]
191         $ns attach-agent $n($node_count) $tcp($i)
192         set tcpsink($i) [new Agent/TCP/FullTcp]
193         incr node_count
194         $ns attach-agent $n($node_count) $tcpsink($i)
195         $ns connect $tcp($i) $tcpsink($i)
196         incr node_count
197         $tcp($i) set fid_ 1
198         $tcpsink($i) listen
199         $tcp($i) set nodelay_ true
200
201         #Setup a CBR over TCP connection?
202
203         set tcpvoip($i) [new Application/Traffic/CBR]
204         $tcpvoip($i) attach-agent $tcp($i)
205         $tcpvoip($i) set type_ CBR
206         $tcpvoip($i) set packetSize_ $voip_size
207         $tcpvoip($i) set rate_ $voip_rate
208         $tcpvoip($i) set random_ false
209     }
210 }
211
212 proc setup-udp-voip {udp_voip voip_size voip_rate} {
213     #Setup a UDP-VoIP connection
214     global ns n node_count udpvoip
215     for {set i 0} {$i < $udp_voip} {incr i} {
216         #Setup a UDP connection
217         puts "One udp-voip"
218         set udp($i) [new Agent/UDP]
219         $ns attach-agent $n($node_count) $udp($i)
220         set udpsink($i) [new Agent/Null]
221         incr node_count
222         $ns attach-agent $n($node_count) $udpsink($i)
223         $ns connect $udp($i) $udpsink($i)
224         incr node_count
225         $udp($i) set packetSize_ 1500
226         $udp($i) set fid_ 2
227
228         #Setup a CBR over UDP connection
```

```
229         set udpvoip($i) [new Application/Traffic/CBR]
230         $udpvoip($i) attach-agent $udp($i)
231         $udpvoip($i) set type_ CBR
232         $udpvoip($i) set packet_size_ $voip_size
233         $udpvoip($i) set rate_ $voip_rate
234         $udpvoip($i) set random_ false
235     }
236 }
237
238 proc setup-pm {pm_count rate rn} {
239     global node_count n CLIENT SERVER pm defaultRNG rng_ flow_arrive req_size rsp_size
240
241     #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
242     # Setup PackMime I.E. webserver- client clouds
243     #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
244     $defaultRNG seed 0
245     # set rate per PackMime
246     set rate_ [expr [expr $rate * 1.0] / $pm_count]
247
248     for {set i 0} {$i < $pm_count} {incr i} {
249         puts "One pack-mime"
250         set pm($i) [new PackMimeHTTP]
251
252         $pm($i) set-server $n($node_count);    # name $n($node_count) as server
253         incr node_count
254         $pm($i) set-client $n($node_count);        # name $n($node_count) as client
255         incr node_count
256
257     }
258     #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
259     # Setup PackMime Random Variables
260     #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
261
262     # create RNGs (appropriate RNG seeds will be assigned automatically)
263     for {set i 0} {$i < 3} {incr i} {
264         set rng_($i) [new RNG]
265     }
266
267     # create RandomVariables
268     set flow_arrive [new RandomVariable/PackMimeHTTPFlowArrive $rate_]
269     set req_size [new RandomVariable/PackMimeHTTPFileSize $rate_ $CLIENT]
270     set rsp_size [new RandomVariable/PackMimeHTTPFileSize $rate_ $SERVER]
271
272     # assign RNGs to RandomVariables
273     $flow_arrive use-rng $rng_(0)
274     $req_size use-rng $rng_(1)
275     $rsp_size use-rng $rng_(2)
276
277     # set PackMime variables
278     for {set i 0} {$i < $pm_count} {incr i} {
279         $pm($i) set-flow_arrive $flow_arrive
280         $pm($i) set-req_size $req_size
281         $pm($i) set-rsp_size $rsp_size
282         $pm($i) set-rate $rate_;                    # new connections per second
283         $pm($i) set-debug 1
284     }
285
286 }
287
288 proc setup-ftp {ftp_count} {
289     global ns n node_count ftp
290
291     for {set i 0} {$i < $ftp_count} {incr i} {
```

```
292             #Setup a TCP connection
293             puts "One ftp-server"
294
295             set tcpf($i) [new Agent/TCP/FullTcp]
296             $ns attach-agent $n($node_count) $tcpf($i)
297             set tcpfsink($i) [new Agent/TCP/FullTcp]
298             incr node_count
299             $ns attach-agent $n($node_count) $tcpfsink($i)
300             $ns connect $tcpf($i) $tcpfsink($i)
301             incr node_count
302             $tcpf($i) set fid_ 3
303             $tcpfsink($i) listen
304             $tcpf($i) set nodelay_ true
305
306             #Setup a FTP over TCP connection
307             set ftp($i) [new Application/FTP]
308             $ftp($i) attach-agent $tcpf($i)
309             $ftp($i) set type_ FTP
310         }
311     }
312
313     global argv
314     process_args
315     # setup TCP specifications
316     Agent/TCP/FullTcp set segsize_ $opts(segsize)
317     Agent/TCP/FullTcp set window $opts(window)
318     Agent/TCP set segsize_ $opts(segsize)
319     Agent/TCP set window $opts(window)
320
321     # calculate the number of nodes needed
322     set num_node [expr ($opts(packmime-count) + $opts(tcp-voip) + $opts(udp-voip) + $opts(ftp-servers) + 1) * 2]
323
324     # call create topology
325     create_topology  $opts(link-bw) $opts(link-delay) $opts(pre-delay) $opts(post-delay) $opts(queue-method) $opts(queue-length)
326
327     # create tracefile and define what is to be traced
328     set datadir /hom/bentj/Master/ns-project/
329     set outfile res_bw($opts(link-bw))_tcp($opts(tcp-voip))_udp($opts(udp-voip))_voiprate($opts(voip-rate))ftp($opts(ftp-servers))
            _pm($opts(packmime-count))_pmrate($opts(rate))_delays($opts(pre-delay))_($opts(link-delay))_($opts(post-delay))_qm($opts
            (queue-method))_ql($opts(queue-length))_run($opts(run)).tr
330     if {$opts(gzip)} {
331         $ns trace-queue $n(0) $n(1) [open "| grep -v ^- | grep -v ^+ | gzip > $datadir$outfile.gz" w]
332     } else {
333         $ns trace-all [open out.tr w]
334     }
335
336     #function call to create TCP based VoIP
337     if {$opts(tcp-voip)} {
338         setup-tcp-voip $opts(tcp-voip) $opts(voip-size) $opts(voip-rate)
339     }
340
341     #function call to create UDP based VoIP
342     if {$opts(udp-voip)} {
343         setup-udp-voip $opts(udp-voip) $opts(voip-size) $opts(voip-rate)
344     }
345
346     #function call to create FTP servers
347     if {$opts(ftp-servers)} {
348         setup-ftp $opts(ftp-servers)
349     }
350
351     #function call to create PackMimeHTTP instances
352     if {$opts(packmime-count)} {
```

```
353     setup-pm $opts(packmime-count) $opts(rate) $opts(run)
354 }
355
356 proc temp {} {
357 }
358 #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
359 # Flush trace
360 #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
361
362 proc finish {} {
363     global ns pm rng_ req_size rsp_size flow_arrive opts
364     $ns flush-trace
365     delete $ns
366     puts "done..."
367     exit 0
368 }
369
370 #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
371 # Simulation Schedule
372 #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
373
374 #start PackMimeHTTP instances
375 for {set i 0} {$i < $opts(packmime-count)} {incr i} {
376     $ns at 0.0 "$pm($i) start"
377 }
378
379 #start ftp servers at random time indexes between 0 and 3
380 for {set i 0} {$i < $opts(ftp-servers)} {incr i} {
381     set randomizeRNG [new RNG]
382     $randomizeRNG seed 0
383     set randomize_ [new RandomVariable/Uniform]
384     $randomize_ set min_ 0
385     $randomize_ set max_ 3
386     $randomize_ use-rng $randomizeRNG
387     $ns at [$randomize_ value] "$ftp($i) start"
388 }
389
390 #start TCP based VoIP at time index 10.(number of TCP based VoIP)
391 for {set i 0} {$i < $opts(tcp-voip)} {incr i} {
392     $ns at 10.[expr $i/10] "$tcpvoip($i) start"
393 }
394
395 #start UDP based VoIP at time index 11.(number of UDP based VoIP)
396 for {set i 0} {$i < $opts(udp-voip)} {incr i} {
397     $ns at 11.[expr $i/10] "$udpvoip($i) start"
398 }
399
400 #end simulation
401 $ns at [expr $opts(duration) + 1] "finish"
402
403 #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
404 # Start the Simulation
405 #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
406
407 $ns run
```

Source code based on [36].

Bibliography

[1] Ieee std 802.3 - 2005 part 3: Carrier sense multiple access with collision detection (csma/cd) access method and physical layer specifications - section three. *IEEE Standards* (December 2005).

[2] At&t. *http://www.att.com/gen/landing-pages?pid=3308* (January 2008).

[3] Cisco systems inc. *http://www.cisco.com/* (January 2008).

[4] JavaSim. *http://javasim.ncl.ac.uk/* (January 2008).

[5] Nam: Network Animator. *http://www.isi.edu/nsnam/nam/* (January 2008).

[6] Network Simulator 2. *http://nsnam.isi.edu/nsnam/index.php/Main_Page* (January 2008).

[7] Random house unabridged dictionary, random house, inc. 2006. *http://dictionary.reference.com/* (January 2008).

[8] Skype official website. *http://www.skype.com/intl/en/* (January 2008).

[9] Telenor asa. *http://www.telenor.com/* (January 2008).

[10] BASET, S. A., AND SCHULZRINNE, H. An analysis of the skype peer-to-peer internet telephony protocol. *ArXiv Computer Science e-prints* (December 2004).

[11] BERNERS-LEE, T., AND CONNOLLY, D. Hypertext markup language - 2.0. *IETF RFC 1866* (November 1995).

[12] C. GRIWODZ, K. A., AND P. HALVORSEN. Redundant bundling in tcp to reduce perceived latency for time-dependent thin streams. In *Accepted for publication in IEEE Communications Letters* (2008).

[13] FÄRBER, J. Network game traffic modelling. In *NetGames '02: Proceedings of the 1st workshop on Network and system support for games* (New York, NY, USA, 2002), ACM, pp. 53–57.

[14] GRIWODZ, C., AND HALVORSEN, P. The fun of using tcp for an mmorpg. In *Network and Operating System Support for Digital Audio and Video (NOSSDAV 2006)* (May 2006), pp. 1–7.

[15] GRIWODZ, C., AND HALVORSEN, P. Tcp congestion control. *University of Oslo Presented in the course INF5071* (September 2007).

[16] HALVORSEN, PAL, B. G. Implementation and evaluation of late data choice for tcp in linux. In *Multimedia, 2007. ISM 2007. Ninth IEEE International Symposium on* (December 2007), pp. 221–228.

[17] INC., C. S. Voice over ip - per call bandwidth consumption. `http://www.cisco.com/en/US/tech/tk652/tk698/technologies_tech_note09186%a0080094ae2.shtml` (January 2008).

[18] INTERNATIONAL TELECOMMUNICATION UNION (ITU-T). One-way Transmission Time, ITU-T Recommendation G.114, 2003.

[19] IPOQUE GMBH. Internet study 2007. `http://www.ipoque.com/media/internet_studies/internet_study_2007` (September 2007).

[20] J. POSTEL ET AL. Transmission control protocol. *IETF RFC 793* (September 1981).

[21] M. HANDLEY, J. H. G. A. J. R., AND E. SCHOOLER. Sip: session initiation protocol.

[22] P. LEACH, R. J. J. H. L., AND T. BERNERS-LEE. Hypertext transfer protocol – http/1.1. *IETF RFC 2616* (Juni 1999).

[23] POSTEL, J. User datagram protocol. *IETF RFC 768* (August 1980).

[24] POSTEL, J., AND REYNOLDS, J. File transfer protocol (ftp). *IETF RFC 959* (October 1985).

[25] R. FREDERICK, H. S., AND V. JACOBSON. Rtp: A transport protocol for real-time applications. *IETF RFC 3550* (July 2003).

[26] RESEARCH, B. L. I. T. PackMime: Statistical Modeling of Connection Request Variables. `http://stat.bell-labs.com/InternetTraffic/packmime.html` (January 2008).

[27] S. ANDERSEN,, AND A. DURIC,. Real-time transport protocol (rtp) payload format for internet low bit rate codec (ilbc) speechinternet low bit rate codec (ilbc). *IETF RFC 3952* (December 2004).

[28] S. FLOYD, AND V. JACOBSON. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking V.1*, N.4 (August 1993), 397–413.

[29] SAT, B., AND WAH, B. W. Playout scheduling and loss-concealments in voip for optimizing conversational voice communication quality. In *MULTIMEDIA '07: Proceedings of the 15th international conference on Multimedia* (New York, NY, USA, 2007), ACM, pp. 137–146.

[30] SCHULZRINNE, H., AND ROSENBERG, J. Internet telephony: architecture and protocols an ietf perspective. *Computer Networks 31* (1999), 237–255.

[31] SULLIVAN, F. . VoIP future outlook: revenue forecast. `http://telephonyonline.com/voip/metrics/voip_revenue_forecast_021705/%` / (February 2005).

[32] TEAM, T. C. Condor Project. `http://www.cs.wisc.edu/condor/` (January 2008).

[33] V. PADMANABHAN,, A., AND C. HERLEY. Understanding and deconstructing bittorrent performance. *ACM SIGMETRICS* (June 2005).

[34] V. PAXSON,, M., AND W. STEVENS. Tcp congestion control. *IETF RFC 2581* (April 1999).

[35] VILLAMIZAR, C., AND SONG, C. High performance tcp in ansnet. *SIGCOMM Comput. Commun. Rev. 24*, 5 (1994), 45–60.

[36] VISWESWARAIAH, V., AND HEIDEMANN, J. rbp_demo.tcl,v 1.4 1998/10/21. October 1998.

[37] W. KLEIJN, S. A. H. R., AND J. LINDEN. Internet low bit rate codec (ilbc). *IETF RFC 3951* (December 2004).